



What's New?

Forum

[New Posts](#) [Private Messages](#) [FAQ](#) [Calendar](#) [Community](#) [Forum Actions](#) [Quick Links](#)[Forum](#) [Pentesting With Kali](#) [Lab Machines](#) [Public Network](#) [10.11.1.71](#) [Offensive Security's Complete Guide to Alpha](#)[Reply to Thread](#)

Results 1 to 10 of 125

Page 1 of 13 [1](#) [2](#) [3](#) [11](#) ... [Last](#) [»](#)**Thread: Offensive Security's Complete Guide to Alpha**[Thread Tools](#) [Search Thread](#)

05-12-2016, 03:58 PM

#1

**g0tmilk**
Offsec Staff

Join Date: Jun 2011

Posts: 538

Offensive Security's Complete Guide to Alpha

Welcome to Offensive Security's complete guide to "Alpha".

Warning. This thread contains spoilers.**Please note that Alpha cannot be included in your Lab Report****Table of Contents:**

- [Introduction](#)
- [Abstract/Overview](#)
- [Reconnaissance](#)
 - [DNS](#)
 - [Lab Notes/Existing Machines](#)
 - [Offsec-Ninja/IRC Bot Hint](#)
- [Information Gathering](#)
 - [Port Scanning](#)
 - [Services \(Part 1 - SSH\)](#)
 - [Services \(Part 2 - HTTP\)](#)
 - [Web Application \(Part 1 - Main\)](#)
 - [Web Application \(Part 2 - Hidden\)](#)
 - [Vulnerabilities vs Exploits vs CVEs](#)
 - [SearchSploit \(Part 1\)](#)
 - [Web Application \(Part 3 - \[SPOILER\]\)](#)
 - [SearchSploit \(Part 2\)](#)
 - [\[SPOILER\]](#)
 - [Web Scanners](#)
- [Limited Shell](#)
 - [Exploit #1 - Manually \(Part 1 - PoC\)](#)
 - [Exploit #1 - Manually \(Part 2 - Remote Shell\)](#)
 - [Exploit #1 - Manually \(Part 3 - Bash Trick\)](#)
 - [Exploit #2 - Exploit-DB](#)
 - [Exploit #3 - Metasploit](#)
- [Privilege Escalation](#)
 - [Information Gathering \(Part 1 - OS\)](#)
 - [Information Gathering \(Part 2 - Running Processes\)](#)
 - [Information Gathering \(Part 3 - Installed Packages\)](#)
 - [Information Gathering \(Part 4 - Installed Programs\)](#)
 - [Information Gathering \(Part 5 - Config files\)](#)
 - [Method #1 - \[SPOILER\] \(Part 1 - Setup\)](#)
 - [Method #1 - \[SPOILER\] \(Part 2 - Exploiting\)](#)
 - [Method #2 - \[SPOILER\] \(Part 1 - SSH\)](#)
 - [Method #2 - \[SPOILER\] \(Part 2 - SU\)](#)
- [Post Exploitation](#)

Last updated: 2017-Sept-12

Last edited by g0tmilk; 09-12-2017 at 12:13 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#)[Reply With Quote](#)

05-12-2016, 04:09 PM

#2



g0tmi1k ◉
Offsec Staff

Join Date: Jun 2011

Posts: 538



Abstract/Overview

Introduction

This is our (Offensive Security) guide to targeting, attacking and completing the machine "Alpha".

It is the first part of a series, with the other machines of "Beta" & "Gamma" (**Coming Soon**).

During all of these machines, we will display how we go about tackling these machines, showing you all our findings & mistakes. Hopefully this will help build up your methodologies and techniques.

Please note, this is a complete walk-through. We will cover everything for Alpha. As a result, if you do not want the machine to be spoiled for you, stop reading now.

Note, we will not accept any other threads on the forum which contain spoilers.

Unlike the other two machines (**Beta & Gamma - Coming Soon**), this is NOT an ex-OSCP exam machine - but an **ex-edb machine (10.11.1.219)**, that we have brought back from the dead!

Normally, we would recommend choosing a target based on the information you know about a machine, rather than going after a specific one.

e.g. The low hanging fruit, rather than hunting for a box or working IP addresses sequentially.

...But we are going to break this rule for this guide.

At this stage, using tools such as nmap/arp-scan/netdiscover would be useful to see all the machines in the subnet which we would have access to, then start port scanning for "key services" (DNS, FTP, HTTP/HTTPS, NetBIOS, SSH/rDesktop/VNC).

We will cover multiple methods of gathering the necessary information to find the vulnerability, from this single issue use three different exploits in order to get a remote shell on the machine. To finish the guide off, two different vulnerabilities to get the highest level of privilege on the system, root.

Abstract/Overview

This machine is vulnerable to the "shellshock" exploit, via Apache's CGI module. The web application (BigTree CMS) is a decoy. We cover two methods to escalate privileges, either by targeting OSSEC (which is meant to protect the OS)! Or by re-trying the MySQL credentials to the non-root user on the box and then sudo'ing to root.

Please note: There may be other methods of getting local access and techniques to acquire a root shell which may be discovered at a later date (*aka undocumented solution in this guide*).

Last edited by g0tmi1k; 09-27-2016 at 10:05 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#)[Reply With Quote](#)

05-12-2016, 04:19 PM

#3



g0tmi1k ◉
Offsec Staff

Join Date: Jun 2011

Posts: 538



Reconnaissance

DNS

The very first thing is to locate this machine (Alpha) in the network.

The easier way to-do this is by completing the course material (**cough* the DNS exercise - because we did do that, right? *cough**). 😊

As this is a guide to Alpha (and not the course material), we will retract certain bits of information here.

Code:

```
root@kali:~# for ip in $(cat /dev/tcp/10.11.1.71 22) ; do echo $ip; done | column -t
...SNIP...
71.1.11.10.in-addr.arpa    name = alpha.thinc.local.
...SNIP...
root@kali:~#
```

```
root@kali:~# for ip in $(cat /dev/tcp/10.11.1.71 22) ; do echo $ip; done | column -t
8.11.11.10.in-addr.arpa    name = alpha.thinc.local.
9.11.11.10.in-addr.arpa    name = alpha.thinc.local.
10.11.11.10.in-addr.arpa   name = alpha.thinc.local.
11.11.11.10.in-addr.arpa   name = alpha.thinc.local.
12.11.11.10.in-addr.arpa   name = alpha.thinc.local.
13.11.11.10.in-addr.arpa   name = alpha.thinc.local.
14.11.11.10.in-addr.arpa   name = alpha.thinc.local.
15.11.11.10.in-addr.arpa   name = alpha.thinc.local.
16.11.11.10.in-addr.arpa   name = alpha.thinc.local.
17.11.11.10.in-addr.arpa   name = alpha.thinc.local.
18.11.11.10.in-addr.arpa   name = alpha.thinc.local.
19.11.11.10.in-addr.arpa   name = alpha.thinc.local.
20.11.11.10.in-addr.arpa   name = alpha.thinc.local.
21.11.11.10.in-addr.arpa   name = alpha.thinc.local.
22.11.11.10.in-addr.arpa   name = alpha.thinc.local.
23.11.11.10.in-addr.arpa   name = alpha.thinc.local.
24.11.11.10.in-addr.arpa   name = alpha.thinc.local.
25.11.11.10.in-addr.arpa   name = alpha.thinc.local.
26.11.11.10.in-addr.arpa   name = alpha.thinc.local.
27.11.11.10.in-addr.arpa   name = alpha.thinc.local.
28.11.11.10.in-addr.arpa   name = alpha.thinc.local.
29.11.11.10.in-addr.arpa   name = alpha.thinc.local.
30.11.11.10.in-addr.arpa   name = alpha.thinc.local.
31.11.11.10.in-addr.arpa   name = alpha.thinc.local.
32.11.11.10.in-addr.arpa   name = alpha.thinc.local.
33.11.11.10.in-addr.arpa   name = alpha.thinc.local.
34.11.11.10.in-addr.arpa   name = alpha.thinc.local.
35.11.11.10.in-addr.arpa   name = alpha.thinc.local.
36.11.11.10.in-addr.arpa   name = alpha.thinc.local.
37.11.11.10.in-addr.arpa   name = alpha.thinc.local.
38.11.11.10.in-addr.arpa   name = alpha.thinc.local.
39.11.11.10.in-addr.arpa   name = alpha.thinc.local.
40.11.11.10.in-addr.arpa   name = alpha.thinc.local.
41.11.11.10.in-addr.arpa   name = alpha.thinc.local.
42.11.11.10.in-addr.arpa   name = alpha.thinc.local.
43.11.11.10.in-addr.arpa   name = alpha.thinc.local.
44.11.11.10.in-addr.arpa   name = alpha.thinc.local.
45.11.11.10.in-addr.arpa   name = alpha.thinc.local.
46.11.11.10.in-addr.arpa   name = alpha.thinc.local.
47.11.11.10.in-addr.arpa   name = alpha.thinc.local.
48.11.11.10.in-addr.arpa   name = alpha.thinc.local.
49.11.11.10.in-addr.arpa   name = alpha.thinc.local.
50.11.11.10.in-addr.arpa   name = alpha.thinc.local.
51.11.11.10.in-addr.arpa   name = alpha.thinc.local.
52.11.11.10.in-addr.arpa   name = alpha.thinc.local.
53.11.11.10.in-addr.arpa   name = alpha.thinc.local.
54.11.11.10.in-addr.arpa   name = alpha.thinc.local.
55.11.11.10.in-addr.arpa   name = alpha.thinc.local.
56.11.11.10.in-addr.arpa   name = alpha.thinc.local.
57.11.11.10.in-addr.arpa   name = alpha.thinc.local.
58.11.11.10.in-addr.arpa   name = alpha.thinc.local.
59.11.11.10.in-addr.arpa   name = alpha.thinc.local.
60.11.11.10.in-addr.arpa   name = alpha.thinc.local.
61.11.11.10.in-addr.arpa   name = alpha.thinc.local.
62.11.11.10.in-addr.arpa   name = alpha.thinc.local.
63.11.11.10.in-addr.arpa   name = alpha.thinc.local.
64.11.11.10.in-addr.arpa   name = alpha.thinc.local.
65.11.11.10.in-addr.arpa   name = alpha.thinc.local.
66.11.11.10.in-addr.arpa   name = alpha.thinc.local.
67.11.11.10.in-addr.arpa   name = alpha.thinc.local.
68.11.11.10.in-addr.arpa   name = alpha.thinc.local.
69.11.11.10.in-addr.arpa   name = alpha.thinc.local.
70.11.11.10.in-addr.arpa   name = alpha.thinc.local.
71.1.11.10.in-addr.arpa    name = alpha.thinc.local.
72.11.11.10.in-addr.arpa   name = alpha.thinc.local.
73.11.11.10.in-addr.arpa   name = alpha.thinc.local.
74.11.11.10.in-addr.arpa   name = alpha.thinc.local.
75.11.11.10.in-addr.arpa   name = alpha.thinc.local.
76.11.11.10.in-addr.arpa   name = alpha.thinc.local.
77.11.11.10.in-addr.arpa   name = alpha.thinc.local.
78.11.11.10.in-addr.arpa   name = alpha.thinc.local.
79.11.11.10.in-addr.arpa   name = alpha.thinc.local.
80.11.11.10.in-addr.arpa   name = alpha.thinc.local.
81.11.11.10.in-addr.arpa   name = alpha.thinc.local.
82.11.11.10.in-addr.arpa   name = alpha.thinc.local.
83.11.11.10.in-addr.arpa   name = alpha.thinc.local.
84.11.11.10.in-addr.arpa   name = alpha.thinc.local.
85.11.11.10.in-addr.arpa   name = alpha.thinc.local.
86.11.11.10.in-addr.arpa   name = alpha.thinc.local.
87.11.11.10.in-addr.arpa   name = alpha.thinc.local.
88.11.11.10.in-addr.arpa   name = alpha.thinc.local.
89.11.11.10.in-addr.arpa   name = alpha.thinc.local.
90.11.11.10.in-addr.arpa   name = alpha.thinc.local.
91.11.11.10.in-addr.arpa   name = alpha.thinc.local.
92.11.11.10.in-addr.arpa   name = alpha.thinc.local.
93.11.11.10.in-addr.arpa   name = alpha.thinc.local.
94.11.11.10.in-addr.arpa   name = alpha.thinc.local.
95.11.11.10.in-addr.arpa   name = alpha.thinc.local.
96.11.11.10.in-addr.arpa   name = alpha.thinc.local.
97.11.11.10.in-addr.arpa   name = alpha.thinc.local.
98.11.11.10.in-addr.arpa   name = alpha.thinc.local.
99.11.11.10.in-addr.arpa   name = alpha.thinc.local.
100.11.11.10.in-addr.arpa  name = alpha.thinc.local.
```

So we can see that **10.11.1.71** is **alpha.thinc.local**.

Lab Notes/Existing Machines

So we now have an IP address (10.11.1.71) and hostname (alpha.thinc.local). At this point, it would be a good time to check our lab notes (made up from information gathered from all the other machines - pre and post exploitation). e.g. Is there any mention of this machine on any network service we can reach? Or are there any personal files/filenames/contents that relate to Alpha? ...As it would be spoiling any other machine(s) - **we made sure this target does not require any others to complete it.**

Offsec-Ninja/IRC Bot Hint

Another thing we can do after we have found out the hostname for a machine, is check the **#Offsec IRC bot (Offsec-Ninja)**. These clues here are not often "directly" useful. Some times its amusing quotes, other times its references to the machine which you may only understand AFTERWARDS. But sometimes... you may get lucky! For how to connect to the channel see **our guide here**. Make sure to **register your nickname** to allow you to talk in the channel.

```
< g0tmi1k > !alpha
<@Offsec-Ninja> g0tmi1k_: alpha is Heroes in a half shell, turtle power.
```

```
[08:13:12] <g0tmi1k_> !alpha
[08:13:12] <@Offsec-Ninja> g0tmi1k_: alpha is Heroes in a half shell, turtle power.
```


Note #1: This is not really useful at this stage, but it will be made clear later on...
Note #2: Some people at this stage may already know of the reference from the franchise "**Teenage Mutant Ninja Turtles**".
Note #3: The IRC hint has been updated to point to this guide.
Last edited by g0tmi1k; 05-23-2017 at 04:38 PM.

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply | Reply With Quote

05-12-2016, 04:43 PM

#4



g0tmi1k Offsec Staff

Join Date: Jun 2011

Posts: 538

Information Gathering
Port Scanning

We are going to start off by...**reverting the machine!**

We haven't got a clue what state the machine is in currently and we do not want to miss anything at this stage. A student may of already exploited a core service, and the vulnerability kills the service, closing the port and we wouldn't be aware - we see *this daily*.

Once the machine has successfully been reverted, we'll do a very quick port scan, then perform a complete scan afterwards. This allows us to start to get an idea and feel for the machine straight away without having to wait about for nmap to complete - else we may have to change up how we are scanning the machine.

#1 - Light Scan

The quick scan (not using any of nmap's inbuilt scripting engine or features) will do the "10 most common ports" (The sorting order of ports is based on nmap's finding over the years).

Code:

```
root@kali:~# nmap 10.11.1.71 --top-ports 10 --open

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:14 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
Not shown: 8 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
root@kali:~#
```

```
root@kali:~# nmap 10.11.1.71 --top-ports 10 --open

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:14 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
Not shown: 8 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
root@kali:~#
```

Two ports open! They are the default ports for **SSH (TCP 22)**, and **HTTP (TCP 80)**.

Note, We haven't confirmed the services behind the ports - just the default ports value are open (**cough* It would be sneaky thing for a system administrator/Offsec to mix up the services *cough**).

Because port 22 is open, defaulting to the SSH service, this hints the target could be *nix based (it is possible SSH is installed on Windows, however it's not "common" at the time of writing - as it could change with Windows 10 in a few years' time...) - and add on the fact we didn't see TCP 3389 being open (Windows RDP) , *nix very often uses VNC instead - which is on a different port.

#2 - Heavy Scan

Now, we can start doing a complete scan (TCP 1 - 65,535), by using "-p-", which is a lot more network 'heavy' (so it's going to take longer).

We are also going to start grabbing the service banners, based on the default service port, by doing "-sV".

It is possible to get nmap to show its justification for its results by doing "--reason".

...and we haven't altered our DNS value (/etc/resolv.conf) to use the PWK lab network, so lets manually use a different value (removed to not spoil the course material, as it is an exercise to find it!)

Note, we didn't use "-A" (which doesn't stand for "all") option, as it makes the output too complex for the time being (as well as make the scan take longer to complete).

Code:

```
root@kali:~# nmap 10.11.1.71 -p- -sV --reason --dns-server [RETRACTED]

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:30 EDT
Nmap scan report for alpha.thinc.local (10.11.1.71)
Host is up, received arp-response (0.16s latency).
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64  OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http?    syn-ack ttl 64
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 663.20 seconds
root@kali:~#
```

```

root@kali:~# nmap 10.11.1.71 -p- -sV --reason --dns-server 10.11.1.1
Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:30 EDT
Nmap scan report for alpha.thinc.local (10.11.1.71)
Host is up, received arp-response (0.16s latency).
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64  OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http?    syn-ack ttl 64
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 663.20 seconds
root@kali:~#

```

As this is going to take "a while to complete" (You can see the scan took over 10 minutes to complete - the light scan, less than a second), we could risk starting to **lightly** use/poke at the target at the same time (or we can use this time to look at another machine, tweak/update our notes, make a drink, or **get on with any other work** etc.).

We don't really want to add too much to the network traffic or the target's system load, as it may make the port scan result inaccurate.

...and if the port scan are incorrect, it's going to upset everything that follows (*We have seen students spend days failing because of incorrect information*).

This is because port scanning is the first thing we do directly to the target - everything after all depends on its results (aka **port scanning is THE essential core stage that we cannot afford to be incorrect** - *which links nicely back to reverting a machine before scanning!*).

So making a few requests to services would be acceptable (**as a normal end user would**). We don't want to start brute forcing services (like a hacker would), or cause any issues/errors (like a hacker may do) as that may trigger some type of protection and block our IP address...

So the port scan finishes, and we only see the two ports that we already knew about. There isn't anything hiding on a sneaky higher port (**cough* that would be mean of us, right? *cough**). We can now start thinking about looking at the services behind the ports...

#3 - Other Port Scan Types

So far we have only touched on TCP ports. **Don't forget about UDP** (**cough* because we haven't *cough**).

Nmap is able to scan for UDP services, however if you thought TCP was slow...

There are other tools out there which are able to perform a port scan (that isn't powered by nmap). One of them being "unicornscan".

I personally find this to be much quicker than nmap (in general), but it doesn't have nmap's powerful scripting engine. One of the advantages of it is the options, control and power you have using it, but the down side to this, unicornscan is slightly more "confusing" to use.

A student of ours, superkojiman, has made a wrapper (onetwopunch - <https://github.com/superkojiman/onetwopunch>) which merges the advantages of unicornscan's speed and nmap's scripts. However, this can be something you research in your own time 🤔.

Side Note: Scanning Multiple Targets At Once & Post Exploiting

We do not recommend scanning a "large amount" of targets at once for various reasons.

Whilst it is "do-able", depending on your network connection (speed and how stable it is - note the VPN uses UDP and not TCP), you will be waiting "a while" (depending on what scanning options you use).

When scanning over a range of targets, nmap will behave by treating all the machines equally. If a machine has a firewall enabled that slows nmap down (or another student attacking/reverting), nmap will then slow down all the other machines to match the same speed as the slowest machine, thus taking longer to complete.

Have you ever seen: "Increasing send delay for [IP] from 0 to 5 due to max successful try/no increase to 4" before?

Nmap offers a wide variety of scanning options, so it's highly recommend to check the man page to get a deeper understanding of the tool. **A few** options to look into are:

- --max-retries
- --max-scan-delay
- --defeat-rst-ratelimit
- reduce the amount of ports you're scanning at once (--top-ports 100, rather than the default 1000 or every port).
- don't use any scripts (or really limit the amount used).
- ...OR bash script a for loop (**cough* like in the course materials *cough**).

Alternatively finding a machine with nmap pre-installed on it - therefore you can scan inside the network, removing a possible bottleneck (your ISP). There are various machines over multiple subnets with nmap on them - lazy system administrators forgetting to remove tools or using tools against themselves.

...but at this stage, you will have no idea what machines these are - but it's something to keep in mind 🤔.

Note, it's NOT recommend to install nmap (or any other tools) on target machines. A reason for this is because if any other student reverts the machine - you would lose it. Plus, it not a "stealthy" option and in a "real life pentest" this may be out of scope (depends on how you're approaching the PWK labs - as there isn't a right or wrong way).

Last edited by g0tmi1k; 07-22-2016 at 04:39 PM.

[Reply](#)[Reply With Quote](#)

05-13-2016, 03:43 PM

#5

**g0tmilk**
Offsec Staff

| | |
|------------|----------|
| Join Date: | Jun 2011 |
| Posts: | 538 |



Information Gathering Services

Based on the nmap report, we can only see 2x TCP ports open, 22 (Defaults to SSH) & 80 (Defaults to HTTP). By checking each port, we can **stop ourselves from getting "tunnel vision"**, by spending a long time going down a rabbit hole only to find it's a "dead end service".

TCP 22 (Default port for SSH)

The SSH protocol itself is "tried and tested" services as it has been around since 1995 (SSH2 was in 2006). The most common service for *nix is "OpenSSH" (started in 1999). It doesn't mean it's without any (publicly known) issues. You can see for yourself on the [CVEDetails.com page](#) (**cough* bookmark this site *cough**)

As a result, **it's not seen as a 'low hanging fruit' attack vector**, as unless something is seriously misconfigured in the SSHD configuration (or if there is a ssh backdoor/rootkit!) the chance of getting a shell out of the box is unlikely. The services often gets brute forced (**cough* it's best to have already gathered a list of usernames beforehand as well as using a few default usernames *cough**), however depending on how the service is configured you may need to use a "private key" (rather than password based) to access the box. You may or may not get a password prompt (depends on how THAT machine is setup), even if it only accepts keys.

However, we may still be able to use it to get some information about the target:

- SSH package version - Might be able to find the OS and version.
- SSH key fingerprint - Has the key been re-used somewhere (Another machine? Same machine, just another port/service?)
- SSH banner - Any text (if at all) before the password prompt (often get legal warnings about connecting to it)

SSH package version

So let's use netcat to connect to the port (it will hang, so we need to kill it once we have our information):

Code:

```
root@kali:~# nc -nv 10.11.1.71 22
(UNKNOWN) [10.11.1.71] 22 (ssh) open
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2
^C
root@kali:~#
```

```
root@kali:~# nc -nv 10.11.1.71 22
(UNKNOWN) [10.11.1.71] 22 (ssh) open
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2
^C
root@kali:~#
```

So we can see the target OS is "**Ubuntu**", using "**OpenSSH v6.6**" (and package is 2ubuntu2). Using this, we can look it up on [Ubuntu's website](#). So using this information, there is a good chance the target is **Ubuntu 14.04 LTS**.

The screenshot shows a web browser window with the URL `packages.ubuntu.com/search?keywords=openssh-server`. The page title is "Exact hits" and the package name is "Package openssh-server". Below this, there is a list of Ubuntu versions and their corresponding architectures for the `openssh-server` package. The list includes versions like `precise (12.04 LTS)`, `precise-updates`, `trusty (14.04 LTS)`, `trusty-updates`, `wily`, `wily-updates`, `xenial`, `xenial-updates`, and `yakkety`. Each entry specifies the architecture (e.g., `amd64`, `arm64`, `armhf`, `i386`, `powerpc`, `ppc64el`, `s390x`) and the package name.

SSH key fingerprint

So when you connect to a SSH service for the first time, SSH will prompt you "do you trust this key"?:

Code:

```
root@kali:~# ssh root@10.11.1.71
The authenticity of host '10.11.1.71 (10.11.1.71)' can't be established.
ECDSA key fingerprint is SHA256:AibCWx1KvdJmNHd3KVsYksWtveJPdLZAshMIChsTeHE.
Are you sure you want to continue connecting (yes/no)?
```

```
root@kali:~# ssh root@10.11.1.71
The authenticity of host '10.11.1.71 (10.11.1.71)' can't be established.
ECDSA key fingerprint is SHA256:AibCWx1KvdJmNHd3KVsYksWtveJPdLZAshMIChsTeHE.
Are you sure you want to continue connecting (yes/no)?
```

Now what happens if you see multiple SSH services on different ports which have the same key? What could it mean if they are different? Why would you see the same key on another box? All questions to think about... As this is not the case here, we will not answer that ☹️ (*cough* but it is in the labs *cough*).

On this subject: A useful resource ~ <https://github.com/rapid7/ssh-badkeys>

SSH banner

Let's go ahead and continue off from the previous command and accept the key:

Code:

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.11.1.71' (ECDSA) to the list of known hosts.
root@10.11.1.71's password:^C

root@kali:~#
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.11.1.71' (ECDSA) to the list of known hosts.
root@10.11.1.71's password:

root@kali:~#
```

So there wasn't any text above the password prompt.

But we DO get a password prompt, so the machine may accept SOME users with a password, rather than keys (or both!).

Example of a banner (able to get some information from it too - domain name!).

Nmap Scripts

We can use nmap to help us out. First, let's check what scripts we have (based on our nmap version):

Code:

```
root@kali:~# ls -lh /usr/share/nmap/scripts/*ssh*
-rw-r--r-- 1 root root 5.6K Mar 31 08:51 /usr/share/nmap/scripts/ssh2-enum-algos.nse
-rw-r--r-- 1 root root 16K Mar 31 08:51 /usr/share/nmap/scripts/ssh-hostkey.nse
-rw-r--r-- 1 root root 1.5K Mar 31 08:51 /usr/share/nmap/scripts/sshv1.nse
root@kali:~#
```

```
root@kali:~# ls -lh /usr/share/nmap/scripts/*ssh*
-rw-r--r-- 1 root root 5.6K Mar 31 03:51 /usr/share/nmap/scripts/ssh2-enum-algos.nse
-rw-r--r-- 1 root root 16K Mar 31 03:51 /usr/share/nmap/scripts/ssh-hostkey.nse
-rw-r--r-- 1 root root 1.5K Mar 31 03:51 /usr/share/nmap/scripts/sshv1.nse
root@kali:~#
```

So by using "-sV" and "--script=ssh-hostkey", we can automate the banner grabbing as well as key fingerprints.

Code:

```
root@kali:~# nmap 10.11.1.71 -p 22 -sV --script=ssh-hostkey
...SNIP...
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 1024 72:b5:55:80:1b:24:d6:f3:bf:a5:c5:98:1b:01:03:90 (DSA)
...SNIP...
root@kali:~#
```

```
root@kali:~# nmap 10.11.1.71 -p 22 -sV --script=ssh-hostkey

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:51 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 1024 72:b5:55:80:1b:24:d6:f3:bf:a5:c5:98:1b:01:03:90 (DSA)
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 93.63 seconds
root@kali:~#
```

Summary

Recommend SSH brute force tools: A **custom wordlist** for the target (using another vulnerability or **CeWL/wordhound**), **Hydra** (don't forget about "-e [VALUES]"), **Patator** (Password fuzzer rather than brute force), **Crowbar** (great for brute forcing private keys), **Metasploit's ssh_login**.

Trouble Shooting: Midway Scanning, port closed?

During our poking about, we noticed after running a few commands, the service started to have a delay in response time. Then **SSH completely stopped responding**.

Scanning it again with nmap, we see the port has changed status (its now filtered):

Code:

```
root@kali:~# nmap -p 22 -sV 10.11.1.71
...SNIP...
22/tcp    filtered ssh
...SNIP...
Nmap done: 1 IP address (1 host up) scanned in 2.51 seconds
root@kali:~#
```



```

root@kali:~# nmap -p 22 -sV 10.11.1.71

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:57 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
MAC Address: 00:50:56:89:54:66 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.51 seconds
root@kali:~# █

```

Oh dear! There's a chance another student has altered the box (though slim, as the machine is still up and responding), however what is much more likely is we have **triggered some type of "defense"** on the machine. There's various solutions in order to stop brute force attempts (multiple failed logins over a period of time), e.g. a common one is "fail2ban". We may of just locked ourselves out (could be just THAT port or the complete machine). Either we can wait until this times out (we don't know how long that would be), or revert the machine.
Note: There are various machines with this in place throughout our PWK/OSCP labs. You would not have to wait more than 15 minutes before you would become unbanned.

Last edited by g0tmi1k; 07-21-2016 at 11:09 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#) | [Reply With Quote](#) |

05-16-2016, 11:52 AM

#6



g0tmi1k ◉
Offsec Staff

| | |
|------------|----------|
| Join Date: | Jun 2011 |
| Posts: | 538 |



Information Gathering Services

TCP 80 (Default port for HTTP)

*Note: Warning HUGE CONTENT! This could be a whole course in just this subject. There is no way we can cover **everything** in this post!*

Before we start:

- Web servers (e.g. Apache/Nginx/IIS) are **not the same** as web applications (e.g. Wordpress/Joomla).
 - Web applications may talk to other services on the OS (such as a database - MySQL/MSSQL)
 - E.g. End User <-> Web Server <-> Web Application <--> Database
 - *And the terms: The Internet (e.g. infrastructure) is not the same as WWW (World Wide Web - web applications)*
- Web servers (e.g. Apache/Nginx/IIS) are **not the same** as web services (e.g. html/json/xml).
- Web servers (e.g. Apache/Nginx/IIS) may have multiple technologies powering them (e.g. PHP/ASP) *via handlers*.
 - These server side technologies are executed on the target directly, not the end users browser (like Javascript).
 - These server side technologies often have more issues with them, as they have a lot more different moving parts, as they need to render/process/execute code written by end users (and the end users code is a whole other set of issues).
- Web servers run on the port (e.g. TCP 80). There can only be a single web server.
 - However, there may be multiple web applications running on the web server.
 - Some may be "hidden". E.g. not linked/clickable from the landing page. Therefore you need to know the URL to go to.
 - e.g. A common hidden web application is PHPMyAdmin.
- Web servers may have multiple "modules" (e.g. Apache) loaded, that expand their functionality (and also be misconfigured!)
 - E.g. /service-status or "index of /" as well as SSL/TLS.

Web servers are found **everywhere** now (examples: traditionally GUI desktop applications moving to a web UI, CLI tool developers using a web UI, or mobile applications just being a browser pointing to selected web page, and embedded device hardware to control network hardware).

Web servers themselves are "dumb", as they only serve/display out what is sent to them. They do not do any processing or rendering themselves.

The common services you will see is **Apache** (both on Windows & *nix), **Ngnix** (Easier on *nix, but there is a Windows port), and **Internet Information Services** (IIS - Windows only... *for the time being!*).

Each of them have had (publicly known) issues over the years (**Apache**, **Nginx**, **IIS**) and they have also had their share of "big" vulnerabilities (which have public exploits - **Apache**, **Nginx**, **IIS**). However, these three services are much more "stable" because they have been around for so long (**older** the version, the more issues - **cough* so it's always worth a checking to see if there is something *cough**). However, there are other web servers other than these three, which haven't been beaten up over the years (**cough* and you may find a few in the labs *cough**).

On the subject of Apache on Windows, it is common to see Apache installed/used as a "package"/bundle, such as XAMPP and WAMP. This then includes other "useful" services at the same time.

So these services directly may not give you a nice shell straight away, what's behind them MIGHT (server side technologies, Web Application, Database). What they are great with is getting information about the target from. Most web servers are "public", allowing anyone to access them.

However, it is worth noticing the different in basic/digest/HTTP authentication (*aka when you get a popup when trying to access a URL*) which happens on the web server, whereas there may be authentication in the web application (*e.g. http forms*). Some times the authentication is done incorrectly, and only the 'default/landing' page has a password prompt, but if you knew/guessed a URL - you may still be able to access it...

Because of the range/scale of what the service allows access to, it's often one of the first services we start probing at. Depending on the web application(s) we can access it may be the low hanging fruit. Until we access the service, we don't know what's behind it.

However, before jumping into the web application, let's check the headers from the web server and default landing page:

Code:

```
root@kali:~# curl -i 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:39:48 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html
```

```
root@kali:~#
```

```
root@kali:~# curl -i 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:39:48 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html
```

```
root@kali:~#
```

So straight away:

- Our request is being redirected (HTTP 302) to: "**site/index.php/**" (Notice how it is two subfolders deep. What is in just **/site/**?)
- The web server appears to be "**Apache 2.4.7**" on **Ubuntu** (which matches what **we know from SSH**)
- Good chance of there being **PHP** on the server: "**v5.5.9**" as well as in the URL redirect: "**site/index.php/**"

Summary

So what we know this far:

- IP: 10.11.1.71 (DNS: alpha.thinc.local)
- Ports: TCP 22, TCP 80 (Might have UDP)
- OS: Ubuntu (Possibly 14.04 - Trusty Tahr)
- Services & Applications:
 - OpenSSH 6.6 - Requires authentication. Might need to brute force it.
 - Apache 2.4.7 & PHP 5.5.9 - No credentials required to access it. **Best bet for the entry point** (unless there's another machine dependence).
- Options left (in order of priority)
 - Explore the web application.
 - Search for vulnerabilities in the known services & applications.
 - Brute force SSH with common & weak credentials.

Last edited by g0tmi1k; 11-22-2016 at 04:41 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-16-2016, 02:47 PM

#7

**g0tmilk**
Offsec Staff

Join Date: Jun 2011

Posts: 538



Information Gathering

Web Application (Main)

Web Application (HTML)

Time to put our "end user" hat on again. What we mean by this is "**use the application, don't try and break it**". What information can you gather by just clicking about and reading what's on screen.

First thing, let's follow the redirect.

Code:

```
root@kali:~# curl -i -L 10.11.1.71
HTTP/1.1 302 Found
...SNIP...
HTTP/1.1 200 OK
...SNIP...
Set-Cookie: PHPSESSID=f81qqelcrdio83ikmumnpabe3; path=/
...SNIP...
Content-Length: 6845
Content-Type: text/html

< !DOCTYPE html>
< html lang="en">
  < head>
    < meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    < meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
    < meta name="keywords" content="" />
    < meta name="description" content="" />

    < title>Trees of Large Sizes< /title>

    < link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all">
    < link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
  ...SNIP...
root@kali:~#
```

```
root@kali:~# curl -i -L 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:42:45 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html

HTTP/1.1 200 OK
Date: Mon, 16 May 2016 22:42:45 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Set-Cookie: PHPSESSID=f81qqelcrdio83ikmumnpabe3; path=/
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
Vary: Accept-Encoding
Content-Length: 6845
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
    <meta name="keywords" content="" />
    <meta name="description" content="" />

    <title>Trees of Large Sizes</title>

    <link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
    <link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
```

It's a valid web application (rather than a default welcome message). So we can see the raw HTML code, which makes up the web page (because this isn't processed on the remote server - unlike PHP code).

Web Application (GUI)

Let's now start up a GUI web browser to see what the page looks like (as a end user).

Trees of Large Sizes

10.11.1.71/site/index.php/

Search

Most Visited

Offensive Security

Kali Linux

Kali Docs

Kali Tools

Exploit-DB


Aircrack-ng

TREES OF LARGE SIZES

TREE BLOG

GLOSSARY

ABOUT



Sequoia

Sequoia is a genus of redwood coniferous trees in the Sequoioideae subfamily, of the Cupressaceae family. The only extant species of the genus is the Sequoia sempervirens in the Northern California coastal forests ecoregion of...

[READ MORE](#)

Photo Credit



By Jos., GFDL or CC-BY-3.0, via Wikimedia Commons

[LEARN MORE](#)

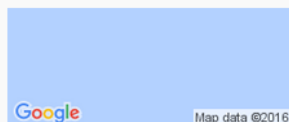
Mr. T Can Chop Down 20 Trees Per Hour

In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down trees during training camp at Fort McCoy in Wisconsin, but did not tell him how many trees, so Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked major superseded the sergeant's orders.

Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The incident is now referred to as "The Lake Forest Chain Saw Massacre"

CONTACT

BigTree CMS
2026 East Lombard Street
Baltimore, MD 21231
support@bigtreecms.org



ABOUT THE BIGTREE SAMPLE SITE

This site is distributed as a technical demonstration of the open source software product BigTree CMS. All content and photographs are sourced from Wikipedia and Wikimedia Commons and are subject to the distribution rights of their respected licenses. We are not tree experts nor do we intend for this site to be used as a factual reference. The design elements...

ACCOUNTS

 **FACEBOOK**
[LIKE US ON FACEBOOK](#)

 **TWITTER**
[FOLLOW US ON TWITTER](#)

and links to other pages/domains.

Web Application (Internal & External Links)

We can't see any HTML comments in the code, so lets now quickly check the title tag (as that can have the web application name in it) as well as any links:

Code:

```
root@kali:~# curl 10.11.1.71 -s -L | grep "title|href" | sed -e 's/^[:space:]]*/'
< title>Trees of Large Sizes< /title>
< link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
...SNIP...
< a href="http://10.11.1.71/site/index.php/" class="branding">Trees of Large Sizes< /a>
...SNIP...
< a href="http://en.wikipedia.org/wiki/Sequoia_%28genus%29" class="more" target="_blank">Read More< /a>
...SNIP...
< a href="http://commons.wikimedia.org/wiki/File%3ASequoia_sempervirens_BigSur.jpg" target="_blank" class="cre
...SNIP...
< p>Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more
< p>< strong>BigTree CMS< /strong>< br /> < span>2026 East Lombard Street < br />Baltimore, MD 21231 < /span><
< a href="http://www.facebook.com/BigTreeCMS" class="facebook" target="_blank">Facebook< small>Like us on Face
< a href="http://www.twitter.com/bigtreecms" class="twitter" target="_blank">Twitter< small>Follow us on Twitt
root@kali:~#
```

```
root@kali:~# curl 10.11.1.71 -s -L | grep "title|href" | sed -e 's/^[:space:]]*/'
<title>Trees of Large Sizes</title>
<link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
<link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
<link rel="stylesheet" href="http://10.11.1.71/site/css/ie.css" type="text/css" media="all" />
<a href="http://10.11.1.71/site/index.php/" class="branding">Trees of Large Sizes</a>
<a href="http://10.11.1.71/site/index.php/tree-blog/">Tree Blog</a>
<a href="http://10.11.1.71/site/index.php/glossary/">Glossary</a>
<a href="http://10.11.1.71/site/index.php/about/">About</a>
<a href="http://en.wikipedia.org/wiki/Sequoia_%28genus%29" class="more" target="_blank">Read More</a>
<a href="http://en.wikipedia.org/wiki/Prunus_serrulata" class="more" target="_blank">Read More</a>
<a href="http://en.wikipedia.org/wiki/Tilia_cordata" class="more" target="_blank">Read More</a>
<a href="http://commons.wikimedia.org/wiki/File%3ASequoia_sempervirens_BigSur.jpg" target="_blank" class="credit active">
<a href="http://commons.wikimedia.org/wiki/File:Cherry_tree_blossoms.jpg" target="_blank" class="credit">
<a href="http://commons.wikimedia.org/wiki/File:Linde_von_linn.jpg" target="_blank" class="credit">
<p>Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The
incident is now referred to as "The Lake Forest Chain Saw Massacre"</p>
<a href="http://en.wikipedia.org/wiki/Mr._T" cla
ss="more">Learn More</a>
<p><strong>BigTree CMS</strong><br /> <span>2026 East Lombard Street <br />Baltimore, MD 21231 </span><br /><a href="mailto:support@bigtreecms.org">supp
ort@bigtreecms.org</a></p>

<a href="http://www.facebook.com/BigTreeCMS" class="facebook" target="_blank">Facebook<small>Like us on Facebook</small></a>
<a href="http://www.twitter.com/bigtreecms" class="twitter" target="_blank">Twitter<small>Follow us on Twitter</small></a>
root@kali:~#
```

Web Application (HTML Render)

So thats all great, but let's now see what the web page renders like.

At this point we can start up iceweasel/firefox/chrome to look at the page... instead, let's stick with command line for the time being. Welcome "html2text".

Code:

```
root@kali:~# curl 10.11.1.71 -s -L | html2text -width '99' | uniq
...SNIP...
*** Mr. T Can Chop Down 20 Trees Per Hour ***
In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down
trees during training camp at Fort McCoy in Wisconsin, but did not tell him how many trees, so
Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked
...SNIP...
* Contact *
BigTree CMS
...SNIP...
support@bigtreecms.org
* Accounts *
FacebookLike_us_on_Facebook TwitterFollow_us_on_Twitter
* About the BigTree Sample Site *
This site is distributed as a technical demonstration of the open source software product BigTree
CMS.
...SNIP...
root@kali:~#
```

```

root@kali:~# curl 10.11.1.71 -s -L | html2text -width '150' | uniq
Trees_of_Large_Sizes
Tree Blog Glossary About
1 2 3
**** Sequoia ****
=====
Sequoia is a genus of redwood coniferous trees in the Sequoioideae subfamily, of the Cupressaceae family. The only extant species of the genus is the
Sequoia sempervirens in the Northern California coastal forests ecoregion of...
Read_More
**** Prunus Serrulata ****
=====
Prunus serrulata or Japanese Cherry; also called Hill Cherry, Oriental Cherry or East Asian Cherry, is a species of cherry native to Japan, Korea and
China. It is known for its spring cherry blossom displays and festivals.
Read_More
**** Tilia Cordata ****
=====
Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Etiam porta sem malesuada magna mollis euismod. Curabitur blandit tempus
porttitor.
Read_More
Photo_Credit
Giles_Douglas,_CC-BY-2.0,_via_Wikimedia_Commons
Photo_Credit
Agricultural_Research_Service,_Public_Domain
Photo_Credit
Stefan_Wernli,_CC-BY-SA-2.5,_via_Wikimedia_Commons
By Jos., GFDL or CC-BY-3.0, via Wikimedia Commons
*** Mr. T Can Chop Down 20 Trees Per Hour ***
In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down trees during training camp at Fort McCoy in Wisconsin,
but did not tell him how many trees, so Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked major
superseded the sergeant's orders.
Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The
incident is now referred to as "The Lake Forest Chain Saw Massacre"
Learn_More
* Contact *
BigTree CMS
2026 East Lombard Street
Baltimore, MD 21231
support@bigtreecms.org
* Accounts *
FacebookLike us on Facebook TwitterFollow us on Twitter
* About the BigTree Sample Site *
This site is distributed as a technical demonstration of the open source software product BigTree CMS. All content and photographs are sourced from
Wikipedia and Wikimedia Commons and are subject to the distribution rights of their respected licenses. We are not tree experts nor do we intend for
this site to be used as a factual reference. The design elements...
© Trees of Large Sizes
root@kali:~#

```

At the top of the page, you would see a lot of content (however it has been snipped out), which is all the navigation menus, then it moves onto content. It doesn't "fully" make sense or fit in. Then theres some non English text (removed). Lastly, there's some text at the bottom (in the footer), which is exactly what we are looking for: "About the **BigTree Sample Site** ...SNIP... demonstration of the **open source** software product **BigTree CMS**". So its a sample site (which explains the odd context of content now), using BigTree CMS.

Web Application (Social Networks)

This can also be re-enforced by one of the social network links, twitter (<http://www.twitter.com/bigtreecms>):

BigTree CMS is an open source content management system built on PHP and MySQL. It was created by, and for, user experience and content strategy experts

BigTree CMS (@bigtreecms) | T

BigTree CMS (@bigtr... x +

Twitter, Inc. (US) | https://twitter.com/bigtreecms

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Home Moments



TWEETS 213 FOLLOWING 8 FOLLOWERS 161 LIKES 86

Tweets Tweets & replies Media

BigTree CMS @bigtreecms
BigTree CMS is an open source content management system built on PHP and MySQL. It was created by, and for, user experience and content strategy experts.
bigtreecms.org
Joined March 2010

BigTree CMS @bigtreecms · Apr 27
BigTree 4.2.10 has been released with release notes here:

 b...
Co...
github.com

Web Application (Accessing The Source Code)

So following the social network home page link (<https://www.bigtreecms.org/>), we can click about until we get the source code (<https://github.com/bigtreecms/BigTree-CMS/>). One of the key files we see is called `"/README.md"`.

GitHub - bigtreecms/BigTree-CMS -

GitHub - bigtreecms/... x +

GitHub, Inc. (US) | https://github.com/bigtreecms/BigTree-CMS/

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Personal Open source Business Explore Pricing Blog Support

bigtreecms / BigTree-CMS

Code Issues 23 Pull requests 0 Pulse Graphs

<http://www.bigtreecms.org>

2,950 commits 6 branches

Branch: master New pull request New file Find file H

timbuckingham Release notes.

| | |
|------------------|--|
| core | Fixed \$bigtree["commands"] being incorrect when previewing a pend |
| .gitignore | Fixed a bunch of failed sql escaping -- shouldn't be a real risk due ... |
| README.md | Release notes. |
| bigtree.sql | Wrong revision # in base sql. |
| example-site.sql | Fixed BigTreeAdmin::ungrowl not doing anything. Fixed bad example |
| install.php | Fixed installing without an explicitly port in some hosting systems. |
| license.txt | 4.0 beta 1 Release |

https://raw...r/README.md x +

https://raw.githubusercontent.com/bigtreecms/BigTree-CMS/master/README.md

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

BigTree CMS 4.2
=====

<http://www.bigtreecms.org/>

Licensing

BigTree CMS is publicly licensed under the [GNU Lesser General Public License](http://www.gnu.org/copyleft/lesser.html). If you would like to use BigTree under a different license, please [contact us](mailto:info@fastspot.com).

Contributing

We would love to have the community work with us on BigTree. Guidelines are currently being created for how community contribution contact <contribute@bigtreecms.org>. If you would like to begin developing the BigTree core, follow the process below:

1. Fork it.
2. Create a branch (`git checkout -b 4.0_toms_branch`)
3. Commit your changes (`git commit -am "Fixed My Broken Foot"`)
4. Push to the branch (`git push origin 4.0_toms_branch`)
5. Create an [Issue][1] with a link to your branch

Changelog

4.2.10 Release

- UPDATED: Data parsers can now be used in both CSV reports and filtered view reports (thanks Jordan Mason)
- UPDATED: TinyMCE to 4.3.10 (default config file settings now include the minified version rather than the developer version)

This is important to us as it contains the "Changelog". Using this, we are able to find out the version of the software.

Code:

```
root@kali:~# curl 10.11.1.71/README.md
BigTree CMS 4.0
=====
< http://www.bigtreecms.org/>
```

```
...SNIP...
Changelog
-----

### 4.0.6 Release
...SNIP...
root@kali:~#
```

```
root@kali:~# curl 10.11.1.71/README.md
BigTree CMS 4.0
=====
<http://www.bigtreecms.org/>

Licensing
-----
BigTree CMS is publicly licensed under the [GNU Lesser General Public License](http://www.gnu.org/copyleft/lesser.html).
If you would like to use BigTree under a different license, please [contact us](mailto:info@fastspot.com).

Contributing
-----
We would love to have the community work with us on BigTree. Guidelines are currently being created for how community can
the project. For more information, please contact <contribute@bigtreecms.org>. If you would like to begin developing the
ow:

1. Fork it.
2. Create a branch (`git checkout -b 4.0_toms_branch`)
3. Commit your changes (`git commit -am "Fixed My Broken Foot"`)
4. Push to the branch (`git push origin 4.0_toms_branch`)
5. Create an [Issue][1] with a link to your branch

Changelog
-----

### 4.0.6 Release
- FIXED: Module Designer not setting id columns to UNSIGNED
- FIXED: Failed BigTreeAutoModule::createItem causing empty cache entries (now properly returns false as well)
```

Bingo! So now we know the name and version, **BigTree CMS v4.0.6**.

Summary

- We now have a few options we could do:
- Poke about a bit more, trying to read any (custom?) content - however, as this is a sample site, maybe not a good idea
 - Try and find the admin control panel - however, we don't have a list of user names to try yet. Could try some we have already gotten in the lab or lookup the default value.
 - Try and find any other web applications on the site (e.g. checking /robots.txt or brute forcing URLs) - This will take some time, so its a good idea to kick it off early.
 - Start looking up if theres any vulnerability in the services and applications so far.

So our plan of action, start to check if there's any more web applications on the server (we are not worried about being stealthy in this attack), and then start researching any known issues and vulnerabilities in software.

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply

Reply With Quote

05-17-2016, 08:59 AM

#8



g0tmi1k 
Offsec Staff

| | |
|------------|----------|
| Join Date: | Jun 2011 |
| Posts: | 538 |



Information Gathering

Web Application (Hidden)

Robots(.txt) vs Spiders

A quick check to see if there's anything the system administrator wouldn't want a Internet spider to index:
Note: For common/default values to look/check for: <https://github.com/h5bp/html5-boilerplate>

```
Code:
root@kali:~# curl 10.11.1.71/robots.txt -s | html2text
***** Not Found *****
The requested URL /robots.txt was not found on this server.
=====

Apache/2.4.7 (Ubuntu) Server at 10.11.1.71 Port 80
```

```
root@kali:~#
```

```
root@kali:~# curl 10.11.1.71/robots.txt -s | html2text
***** Not Found *****
The requested URL /robots.txt was not found on this server.
=====
      Apache/2.4.7 (Ubuntu) Server at 10.11.1.71 Port 80
root@kali:~#
```

Useful tool: **parsero**

URL Brute Force (General)

Brute forcing doesn't always mean passwords attacks. Its the process of guessing, by trying certain combinations (either pre-defined from a dictionary/wordlist or trying every possible value, increasing its value each time). We can apply this to URLs too.

There's various tools to help us do this, such as: **DirB** (CLI), **DirBuster** (GUI), **wfuzz** (CLI), **Burp Suite** (GUI), and *my favourite* **Gobuster** (CLI).

Just like password brute forcing, it doesn't matter how "good" a tool is, the key is the wordlist. **If the "magic" value isn't in the wordlist, its not going to be discovered.** And keep in mind the longer the wordlist, the longer the attack will take.

Note: The labs have been designed so you should not be brute forcing anything for more than 30 minutes.

Some of the mentioned tools come with their own wordlists with them (such as DirB & wfuzz) which have commonly found URLs - *and you can mix and match the tools to the wordlists.*

However, there is a dedicated project called "**SecList**" which aims to cover as many general/generic wordlists as possible (for every topic). Its worth having a quick explore:

- DirB - **/usr/share/dirb/wordlists/**
- wfuzz - **/usr/share/wfuzz/wordlist/**
- SecList - **/usr/share/seclists/**

Note: Depending on your Kali version, you may have to install them (apt-get install -y [name])

Code:

```
root@kali:~# gobuster -u http://10.11.1.71/ \
-w /usr/share/seclists/Discovery/Web_Content/common.txt \
-s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
              (DNS support by Peleus @0x42424242)
=====
[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist       : /usr/share/seclists/Discovery/Web_Content/common.txt
[+] Status codes   : 500,200,204,301,302,307,403
[+] Expanded      : true
=====
http://10.11.1.71/.hta (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/cache (Status: 301)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/core (Status: 301)
http://10.11.1.71/custom (Status: 301)
http://10.11.1.71/index.php (Status: 302)
http://10.11.1.71/javascript (Status: 301)
http://10.11.1.71/phpmyadmin (Status: 301)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/site (Status: 301)
http://10.11.1.71/templates (Status: 301)
...SNIP...
root@kali:~#
```

```

root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/common.txt \
> -s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
          (DNS support by Peleus @0x42424242)
=====
[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist        : /usr/share/seclists/Discovery/Web_Content/common.txt
[+] Status codes   : 500,200,204,301,302,307,403
[+] Expanded       : true
=====
http://10.11.1.71/.hta (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/cache (Status: 301)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/core (Status: 301)
http://10.11.1.71/custom (Status: 301)
http://10.11.1.71/index.php (Status: 302)
http://10.11.1.71/javascript (Status: 301)
http://10.11.1.71/phpmyadmin (Status: 301)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/site (Status: 301)
http://10.11.1.71/templates (Status: 301)
=====
root@kali:~#

```

So let's break these values down:

```

/index.php (Status: 302)
/cache (Status: 301)
/core (Status: 301)
/custom (Status: 301)
/javascript (Status: 301)
/phpmyadmin (Status: 301) <-- Could be something.
/site (Status: 301)
/templates (Status: 301)
/.hta, /.htaccess, /.htpasswd (Status: 403) <-- Good chance ".dot files" are not allowed to be accessed directly.
/cgi-bin/ (Status: 403) <-- Could be something.
/server-status (Status: 403) <-- Shame. Might of got some nice information about the machine.

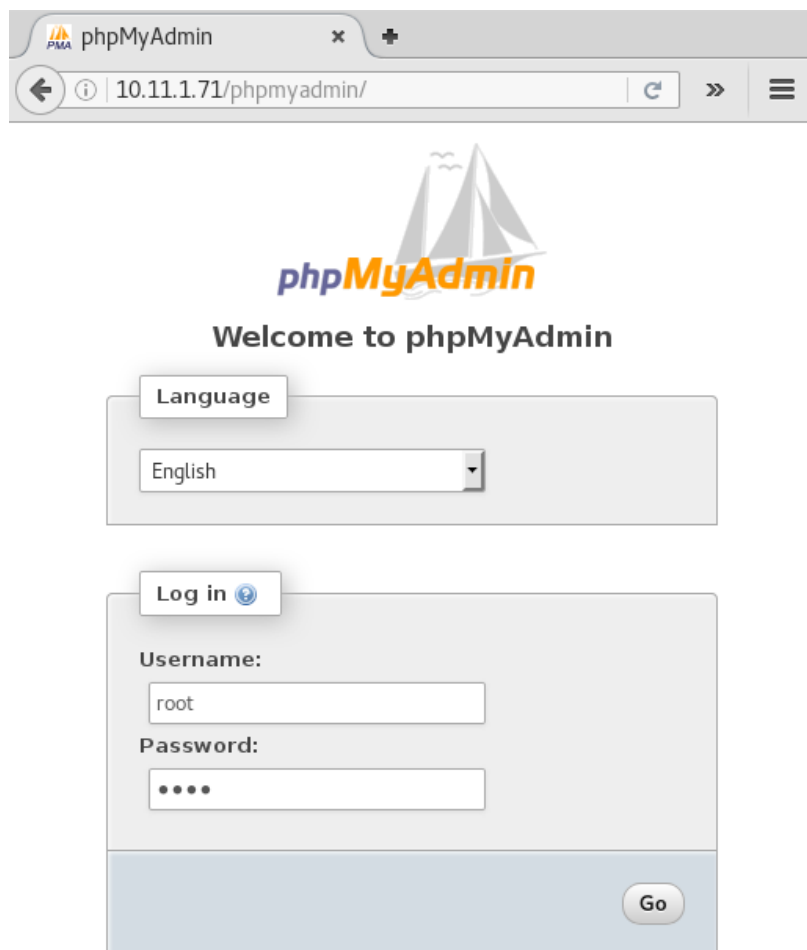
```

Options now:

- Poke around at phpMyAdmin.
- Try a different wordlist. Either/mixture/all:
 - Another general one.
 - One targeted towards: /cgi-bin/.
 - ...We don't have enough (custom) information to generate one towards our target (using **CeWL/wordhound**), as its using a sample page.
- Start researching vulnerabilities and issues in known software.
 - The later we do this, hopefully we will know more about the target and have more to research, increasing the possible attack surface.

phpMyAdmin

So using iceweasel/firefox/chrome, we can navigate to <http://10.11.1.71/phpmyadmin> and see the following:



We manually try a few default values such as:

- **admin** \ "" (blank), **admin** \ **admin**, **admin** \ **password**
- **root** \ "" (blank), **root** \ **root**, **root** \ **password**

However, it wasn't successful. A Google search shows the default passwords depends on how phpMyAdmin was installed and the OS and it's version - we tried every value.

We could start a online password attack on it, however the chance of it being successful is slim. **We'll put a pin it in now, and put it at the end of to "try list" so we will revisit it at a later time (if required).**

URL Brute Force (CGI)

So next on our list, another wordlist to try! As we have already done a wordlist of common values, let's use what we learnt from it and start to get a specific/specialize, by targeting CGI URLs.

All we are going to-do, is re-run the same command as last time, however switch out the wordlist with another one from SecList:

Code:

```
root@kali:~# gobuster -u http://10.11.1.71/ \
-w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
-s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
(DNS support by Peleus @0x42424242)
=====
[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist       : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes  : 204,301,302,307,403,500,200
[+] Expanded      : true
=====
http://10.11.1.71/. (Status: 302)
http://10.11.1.71/index.php?chemin=.%2F..%2F..%2F..%2F..%2F..%2F%2Fetc (Status: 302)
http://10.11.1.71/index.php/123 (Status: 302)
http://10.11.1.71/?mod=node&nid=some_thing&op=view (Status: 302)
http://10.11.1.71/?mod=some_thing&op=browse (Status: 302)
http://10.11.1.71/index.php?file=index.php (Status: 302)
^C
root@kali:~#
```

```

root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
> -s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
          (DNS support by Peleus @0x42424242)
=====

[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist        : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes   : 204,301,302,307,403,500,200
[+] Expanded      : true
=====

http://10.11.1.71/. (Status: 302)
http://10.11.1.71/index.php?chemin=..%2F..%2F..%2F..%2F..%2F..%2F%2Fetc (Status: 302)
http://10.11.1.71/index.php/123 (Status: 302)
http://10.11.1.71/?mod=node&nid=some_thing&op=view (Status: 302)
http://10.11.1.71/?mod=some_thing&op=browse (Status: 302)
http://10.11.1.71/index.php?file=index.php (Status: 302)
^C
root@kali:~#

```

...We killed it before it could complete. This is because we are just been flooded with data that we are not interested in right now (all those redirects - HTTP 302). Let's filter them out for the time being, if we need to, re-scan again with them enabled (we put it on our on "to try later" list).

Useful: [List of HTTP status codes](#)

Code:

```

root@kali:~# gobuster -u http://10.11.1.71/ \
-w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
-s '200,204,403,500' -e
...SNIP...
[+] Status codes : 500,200,204,403
...SNIP...
http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../etc/passwd (Status: 200)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/phpmyadmin/ (Status: 200)
http://10.11.1.71/cgi-bin/admin.cgi (Status: 200)
http://10.11.1.71/cgi-bin/test.cgi (Status: 200)
http://10.11.1.71/cgi-bin/.htaccess (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.old (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.save (Status: 403)
http://10.11.1.71/cgi-bin/.htpasswd (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess~ (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/icons/ (Status: 403)
...SNIP...
root@kali:~#

```

```
root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
> -s '200,204,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
              (DNS support by Peleus @0x42424242)
=====

[+] Mode       : dir
[+] Url/Domain  : http://10.11.1.71/
[+] Threads    : 10
[+] Wordlist     : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes : 200,204,403
[+] Expanded    : true
=====

http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../etc/passwd (Status: 200)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/phpmyadmin/ (Status: 200)
http://10.11.1.71/cgi-bin/admin.cgi (Status: 200)
http://10.11.1.71/cgi-bin/test.cgi (Status: 200)
http://10.11.1.71/cgi-bin/.htaccess (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.old (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.save (Status: 403)
http://10.11.1.71/cgi-bin/.htpasswd (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess~ (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/icons/ (Status: 403)
=====

root@kali:~# █
```

Let's break it down again:

```
/cgi-bin/ (Status: 403) <-- We can't access in the index page... but...
/cgi-bin/admin.cgi (Status: 200) <-- ...we can any page (if we know/guess the address!). Could be something.
/cgi-bin/admin.cgi?list=../../../../../../../../etc/passwd (Status: 200) <-- This might be a LFI vulnerability (if it accepts/uses 'list' as a
input).
/cgi-bin/test.cgi (Status: 200) <-- Could be something.
/phpmyadmin/ (Status: 200) <-- Knew this already for last time.
/cgi-bin/.htaccess, /cgi-bin/.htaccess.old, /cgi-bin/.htaccess.save, /cgi-bin/.htpasswd, /cgi-bin/.htaccess~, /.htpasswd, /.htaccess
(Status: 403) <-- Like last time, good chance "dot files" are not allowed to be accessed.
/server-status (Status: 403) <-- Like last time, Can't use it to get some nice information about the machine.
/icons/ (Status: 403) <-- Not helpful.
```

The take away on it this time around, anything starting with ".hta" is blocked. This has NOT been applied to ".cgi-bin/" URLs, so if you can guess/predict what's on the target, we can access them (just not the index page!).

Summary

We have three hidden URLs to look at:

- <http://10.11.1.71/phpmyadmin/>
- <http://10.11.1.71/cgi-bin/admin.cgi>
- <http://10.11.1.71/cgi-bin/test.cgi>

Last edited by g0tmi1k; 07-22-2016 at 04:34 PM.

[Reply](#) | [Reply With Quote](#)

05-18-2016, 12:19 PM

#9



g0tmi1k ◉
Offsec Staff

| | |
|------------|----------|
| Join Date: | Jun 2011 |
| Posts: | 538 |



Information Gathering Vulnerabilities vs Exploits vs CVEs

A quick overview of a few terms:

- A **vulnerability** is flaw in a system which COULD provide an attacker with a way into the software itself, in a unattended manner.
 - It is not an open door, but a weak door, which MIGHT allow an attacker a way in.

- A **exploit** is the way INTO the system. An attacker turns the vulnerability into a method into the system.
 - An exploit is the tool used to bust down the door - allowing the attacker to walk through the door.
- **0day** means the exploit has been known about for less than a day. So the software authors didn't have any notice/chance to create a patch, to protect from the vulnerability.
 - Someone has found a way to bust down a door without giving the chance to put up any protections, stopping the attack from happening.
- **1day** means the vulnerability is publicly known about, allowing for the software authors to create a patch. However, there isn't yet any public exploit code.
 - Able to protect a door from being busted down even though there isn't yet a known way to open the door.

CVE is a standard, for making a list of vulnerabilities, using a certain naming format and terms. It makes it easier to identity and reference vulnerabilities.

- Able to identity what the issue is.
- A **"feature"** is using the software how it was designed in order to perform an action
 - Such as allowing file uploads on a web site, to share pictures, might also allow for web shells to be uploaded.

Please note:

- Not every vulnerability can be exploited (for various reasons).
- Not every vulnerability has an exploit (for various reasons).
- Not every vulnerability or exploit is "public" (Sometimes are kept "private" so they are not shared with anyone or required to be purchased. Sometimes these vulnerabilities are not even publicly known about!)
- Not every vulnerability has a CVE (for various reasons).
- There are other naming conventions than CVEs to identity issues (such as Microsoft's Bulletins).
- There might be multiple exploits for the same vulnerability (e.g. re-written in different coding languages).
- One exploit might use/target multiple vulnerabilities.
- Exploits may get updated over time (just like any other software)!
- Exploits may affect a range of software versions (depends on the vulnerability, which depends on the code used in the software in the first place).
- It's possible to chain vulnerabilities to create a exploit (Allowing to access/reach places in code which normally wouldn't be accessible).
- Some "Denial of Service (DoS)" exploits are the beginning of creating a PoC exploit. Not every DoS exploit can be converted (goes back to not every vulnerability can be exploited).
- A "Proof of Concept (PoC)" exploit is to demonstrate the vulnerability is exploitable. Depending on the state of the PoC, it may require work in order to reach desired goal (E.g. it displays a pop up alert, rather than executing commands on the target). These are not always stable.
- A weaponized exploit, means the payload will work for everyone/anyone every time, out of the box without any configuration needed. These are stable.
- ...*This is only a quick overview/guide!*

Attack Surface based on Target's Information

Up till now, we really haven't tried to "hack" into the target - more about just being an end user and gathering information about the system (might of just tried a default. Using what we know, we have some information about the software installed on the target. Let's put what we know in order (based on the attack surface chance of being vulnerable):

1. Web Application - BigTree CMS 4.0.6
2. Web Technologies - PHP 5.5.9
3. Web Server - Apache 2.4.7
4. SSH Service - OpenSSH 6.6
5. Database - MySQL (Not sure on the version)
6. OS - Ubuntu (14.04? - not sure on the version)

The justification for this, mainly comes from experience (background knowledge based on issues seen in the past):

- We don't have **direct** access to either the OS or MySQL database, as well as know their version number at this stage - which is why they are at the bottom.
- SSH is known being 'stable' service - (which we know from researching CVEs).
- The web server is relatively modern. There's only a very slim chance there will be a vulnerable issue with it.
- The PHP core itself (not user created code) has had various issues in recent years - so there is a chance it could be vulnerable to *something*
 - however often requires a certain setting/actions, which lowers its possibility we can exploit something.
- The highest chance of there being an exploit, allowing us to get a foot hold into the system, would be in the web application.
 - This is because it is the newest technology so would have had less time to mature due to less developers/pairs of eyes looking and working on the code.
 - It also has the largest attack surface based on it being able to access the database (nothing else which we can interact with can).

Exploit-Database & CVEs

Exploit-Database (sometimes called, Exploit-DB or E-DB) is exactly what it says, a collection of exploits (all of which are free to access). This can be accessed either online via the web site (<https://www.exploit-db.com/>) or offline using the command line tool, **searchsploit**. Using them, it is possible to search for known exploits (not vulnerabilities!) using various terms/criteria, such as software, versions and CVEs. For more information about **Exploit-DB**, [see here](#), else [see here for SearchSploit](#). *Note: Whilst there are other sources for exploits (such as **SecurityFocus** and **PacketStorm**), all the exploits you will require for the labs can be found on Exploit-DB (which is maintained by Offensive Security)!*

Useful tools: **vFeed** (allows searching for known vulnerabilities, not exploits), **searchsploit** (exploit database)

Something to keep in mind, as the exploits hosted on Exploit-DB are submitted from the exploit authors, their exploit title formats may differ slightly. This means it may take multiple different search terms to find the exploit you are looking for (*more about this later*). This is when searching for CVEs is more useful, however it requires researching and knowing of a CVE value before hand...

- Useful CVE sites:
- CVE lookup - <https://www.cvedetails.com/vendor.php> (**Great to see an overview**)
 - CVE information - [https://www.cvedetails.com/cve/\[CVE\]](https://www.cvedetails.com/cve/[CVE])
 - CVE information - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=\[CVE\]](http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE])
 - CVE information - [https://web.nvd.nist.gov/view/vuln/detail?vulnId=\[CVE\]](https://web.nvd.nist.gov/view/vuln/detail?vulnId=[CVE])
 - Depending on the site of the software, it may also be tracking CVEs on its own page (and often has a lot more information about the issue) - e.g. [https://security-tracker.debian.org/tracker/\[CVE\]](https://security-tracker.debian.org/tracker/[CVE])
 - CVE sources - <https://cve.mitre.org/data/refs/index.html>

Last edited by g0tmilk; 11-22-2016 at 04:29 PM.

Reply

Reply With Quote

05-18-2016, 02:35 PM

#10



g0tmilk
Offsec Staff

| | |
|------------|----------|
| Join Date: | Jun 2011 |
| Posts: | 538 |

Information Gathering

SearchSploit (Part 1)

For the purpose of demonstrating searching, let's forget the attack surface ordering which was defined/discuss earlier. This will allow us to show additional *tips* on searching, allowing us to find exploits quicker.

OpenSSH 6.6

Let's see what we can find out about this service.

Code:

```
root@kali:~# searchsploit OpenSSH 6.6
...
```



Great start - 0 results.

We were a little specific with what we searched for, so let's loosen it up, by removing the subversion.

- Example #1: What if the exploit title was called "OpenSSH 6.x" (meaning for every subversion of v6?)?
- Example #2: What if the title is "OpenSSH <= 6.8" (meaning every version of 6.8 and under)

Side Note: This is something to always keep in mind when searching for Linux Kernel exploits!

Code:

```
root@kali:~# searchsploit OpenSSH 6
...
```

```

root@kali:~# searchsploit OpenSSH 6
-----
Exploit Title | Path
-----|-----
OpenSSH/PAM <= 3.6.1p1 - Remote Users Discovery Tool | ./linux/remote/25.c
OpenSSH/PAM <= 3.6.1p1 - Remote Users Ident (gossh.sh) | ./linux/remote/26.sh
Portable OpenSSH <= 3.6.1p-PAM / 4.1-SUSE Timing Attack Exploit | ./multiple/remote/3303.sh
Debian OpenSSH - Remote SELinux Privilege Elevation Exploit (auth) | ./linux/remote/6094.txt
Novell Netware 6.5 - OpenSSH Remote Stack Overflow | ./novell/dos/14866.txt
FreeBSD OpenSSH 3.5p1 - Remote Root Exploit | ./freebsd/remote/17462.txt
OpenSSH <= 7.2p1 - xauth Injection | ./multiple/remote/39569.py
-----
root@kali:~#

```

SearchSploit started searching in the path for "6", however, its not too many results to quickly eye ball.

We soon see there isn't any exploits that would "fit". We are using two lose terms, and there isn't any other way they could be called so we can rule out this one for the time being (*we add to our "to try" list about looking up CVEs and then search for them*).

Onto the next, **Apache 2.4.7**.

Code:

```

root@kali:~# searchsploit Apache 2.4.7
...

```

```

root@kali:~# searchsploit Apache 2.4.7
-----
Exploit Title | Path
-----|-----
Apache 2.4.7 mod_status Scoreboard Handling Race Condition | ./linux/dos/34133.txt
-----
root@kali:~#

```

This might be an exact match for the software and version, however its not "helpful" for us for a few reasons.

1. Its a DoS exploit (based on the path - ./linux/dos/34133.txt) - not really going to help anyone here (its not labeled as a PoC, and not looking to develop a PoC)
2. We know from brute forcing the URL that the page responded with a HTTP 403 request. As the title of the exploit doesn't even hint at bypassing this limitation, its not going to work)

Note: Developing a exploit from a DoS exploit is out of scope for OSCP, as it is NOT in the course material or syllabus.

By using a bit of "grep fu", we are able to remove any DoS exploits in our searches. Example:

Code:

```

root@kali:~# searchsploit Apache 2.4.7 | grep -v '/dos/'
...

```

```

root@kali:~# searchsploit Apache 2.4.7 | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
-----
root@kali:~#

```

Note: We didn't use "grep -v 'dos'", as we want the slashes, hinting its a path. Else we might be filtering out "dos" from the titles (as grep isn't removing based on whole words!)

Time to relax the search terms again, by trying: "searchsploit Apache 2.4" and searchsploit Apache 2.x"

Code:

```

root@kali:~# searchsploit apache 2.4 | grep -v '/dos/'
...
root@kali:~# searchsploit apache 2.x | grep -v '/dos/'
...

```

```

root@kali:~# searchsploit apache 2.4 | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
Apache Tomcat 3.2.3/3.2.4 - Source.JSP Malformed Request Information Disclosure | ./multiple/remote/21490.txt
Apache Tomcat 3.2.3/3.2.4 - Example Files Web Root Path Disclosure | ./multiple/remote/21491.txt
Apache Tomcat 3.2.3/3.2.4 - RealPath.JSP Malformed Request Information Disclosure | ./multiple/remote/21492.txt
Apache HTTP Server <= 2.2.4 - 413 Error HTTP Request Method Cross-Site Scripting Weakness | ./unix/remote/30835.sh
-----
root@kali:~#
root@kali:~# searchsploit apache 2.x | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
-----
root@kali:~#

```

None of these results are a good match for us.

Time for the next software, **PHP 5.5.9**.

This is where the power of bash's command line tools really help out (over using the web interface).

We are wanting to search for PHP's "core", rather than PHP platform based exploits, or filenames with ".PHP" in it (its common in exploits titles to indicate the vulnerable web page). If we were to search like we have been, we would get 87 possible exploits (which will take a short while to look through - however we can do better!).

Code:

```
root@kali:~# searchsploit php 5.x
...
```

```
root@kali:~# searchsploit php 5.x | wc -l
87
root@kali:~#
root@kali:~# searchsploit php 5.x | head
-----
Exploit Title                                                    | Path
-----|-----
| (/usr/share/exploitdb/platforms)
UBB Threads 5.x / 6.x - Multiple Remote File Inclusion Vulnerabilities | ./php/webapps/1843.txt
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit           | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass     | ./windows/local/4553.php
MoinMoin 1.5.x MOIND_ID cookie Bug Remote Exploit                 | ./php/webapps/4957.txt
Battle.net Clan Script <= 1.5.x - Remote SQL Injection Exploit    | ./php/webapps/5597.pl
Joomla 1.5.x - (Token) Remote Admin Change Password Vulnerability | ./php/webapps/6234.txt
root@kali:~#
```

So what we are going to-do, is look at ONLY the exploit titles (by using "-t"), so we forget about the the exploit path (aka the platform):

Code:

```
root@kali:~# searchsploit -t php 5.x
...
```

```
root@kali:~# searchsploit -t php 5.x | wc -l
60
root@kali:~#
root@kali:~# searchsploit -t php 5.x | head
-----
Exploit Title                                                    | Path
-----|-----
| (/usr/share/exploitdb/platforms)
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit           | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass     | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability               | ./php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | ./php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | ./php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability    | ./php/webapps/21230.txt
root@kali:~#
```

That's a few unwanted results gone, but we can do better. Now, let's remove all DoS exploits (just like before).

Code:

```
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/'
...
```

```
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | wc -l
57
root@kali:~#
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | head
-----
Exploit Title                                                    | Path
-----|-----
| (/usr/share/exploitdb/platforms)
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit           | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass     | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability               | ./php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | ./php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | ./php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability    | ./php/webapps/21230.txt
root@kali:~#
```

Now, let's try and remove all ".php" results (we are only after PHP's core, rather than web page ending with .php)..

Code:

```
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
...
```

```
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
-----
Exploit Title | Path
-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | /windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | /windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | /php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | /php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | /php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | /php/webapps/21230.txt
root@kali:~#
```

Wait. That didn't work right. Why didn't that remove what we wanted?

The problem is because of the highlighting. See how we search for 'php', and all the values in red? Now we are wanting to remove "dot php". But to us/end users that looks the same, but that's not how it is on Kali. This is because there's some "invisible" strings used, to perform the highlighting.

Picking on **"PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability"**

Code:

```
user.\[033[1;31m\]php
```

So we can tell searchsploit not to add colour ("**--colour**"), thus making grep work as we want it to!

Code:

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php'
...
```

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | wc -l
19
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
-----
Exploit Title | Path
-----
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | /php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | /php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL_Debug Information Disclosure Vulnerability | /php/webapps/21233.txt
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | /php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | /php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | /php/webapps/22211.txt
root@kali:~#
```

Bingo! Much less!

However, there is a slight mistake in the command. If an exploit has a ".PHP" file extension for the PHP core, it would be removed. We can fix this by adding a trailing space.

Not completely perfect, as we are going to see...

Code:

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php '
...
```

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php ' | wc -l
26
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php '
-----
Exploit Title | Path
-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | /windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | /windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | /php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | /php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL_Debug Information Disclosure Vulnerability | /php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1) | /php/remotes/21264.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (2) | /php/remotes/21265.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (3) | /php/remotes/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | /php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | /php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | /php/webapps/22211.txt
PHP-Nuke 5.x/6.x Web_Links Module - Remote SQL Injection Vulnerability | /php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability | /php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities | /php/webapps/24232.txt
PHP 4.x/5.x - Html_Entity.Decode() Information Disclosure Vulnerability | /php/remotes/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | /php/remotes/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | /php/remotes/29316.py
Battle.net Clan Script 1.5.x - 'index.php' Multiple SQL Injection Vulnerabilities | /php/webapps/32181.txt
Simple PHP Blog 0.5.x - 'search.php' Cross-Site Scripting Vulnerability | /php/webapps/33507.txt
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | /php/remotes/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | /php/webapps/35146.txt
root@kali:~#
```

So we are removing potential exploits! However, we have added a few more false positive values back in.

If we really want to get fancy, we can start to use "regex" (Regular Expressions) to filter all ".php" results *except* for when the

end of the line, ends with ".php"

Code:

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]'
...
```

```
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]' | wc -l
24
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]'

-----
Exploit Title                                                                 Path
-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit                    ./windows/local/4236.php
PHP 5.x - COM functions safe_mode and disable_function bypass             ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability                       ./php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability            ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL Debug Information Disclosure Vulnerability          ./php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (1)     ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (2)     ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (3)     ./php/remote/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability            ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability                     ./php/webapps/22211.txt
PHP-Nuke 5.x/6.x Web Links Module - Remote SQL Injection Vulnerability     ./php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability    ./php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities           ./php/webapps/24232.txt
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability     ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)         ./php/webapps/35146.txt
-----
root@kali:~#
```

...if we really want to get flashy, we can compress the line slightly by removing a pipe.

Code:

```
root@kali:~# searchsploit --colour -t php 5.x | grep -vi '/dos/\\\.php[^\$]'
...
```

But we are not yet out of the woods!

Let's also in a single command look for a "5.x" and "5.5" exploits. Can use either "**grep '5\\.\\(5\\|x\\)'**" or "**grep '5\\.5\\|5\\.x'**"

Code:

```
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\\.\\(5\\|x\\)'
...
```

```
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\\.\\(5\\|x\\)' | wc -l
24
root@kali:~#
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\\.\\(5\\|x\\)'

-----
Exploit Title                                                                 Path
-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit                    ./windows/local/4236.php
PHP 5.x - COM functions safe_mode and disable_function bypass             ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability                       ./php/webapps/12272.txt
PHP Gift Registry 1.5.5 - SQL Injection                                   ./php/webapps/18519.txt
Artiphp CMS 5.5.0 Database Backup Disclosure Exploit                      ./php/webapps/18889.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability            ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL Debug Information Disclosure Vulnerability          ./php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (1)     ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (2)     ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (3)     ./php/remote/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability            ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability                     ./php/webapps/22211.txt
PHP-Nuke 5.5/6.0 AvantGo Module - Path Disclosure Vulnerability           ./php/webapps/22347.txt
PHP-Nuke 5.5/6.0 News Module - Path Disclosure Vulnerability              ./php/webapps/22348.txt
PHP-Nuke 5.x/6.x Web Links Module - Remote SQL Injection Vulnerability     ./php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability    ./php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities           ./php/webapps/24232.txt
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability     ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)         ./php/webapps/35146.txt
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit                         ./multiple/remote/39645.php
-----
root@kali:~#
```

...If you are saying to yourself now "but they both have 24 results" - correct! However, the latter command doesn't have the "header" and "footer" lines. So there is more exploits!

So after all of that with SearchSploit, is there anything? Yes!

After removing the "Windows" exploits (target is Ubuntu, remember?), as well as programs that use PHP in the name (such as "PHP-Nuke", "RapidKill Pro", "Gift Registry" etc), we get...

Code:

```
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\\.\\(5\\|x\\)' | \
grep -vi '/windows/\\|PHP-Nuke\\|RapidKill Pro\\|Gift Registry\\|Artiphp CMS'
...
```

```
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\.\.php[^\$]' | grep -i '5\\.\\(5\\|x\\)' | \
> grep -vi '/windows/\\|PHP-Nuke\\|RapidKill Pro\\|Gift Registry\\|Artiphp CMS'
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1) | ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (2) | ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (3) | ./php/remote/21266.php
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability | ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | ./php/webapps/35146.txt
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit | ./multiple/remote/39645.php
root@kali:~#
```

Now we can manually remove the lower versions (e.g. 5.4.x), we have the following candidates:

PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1), (2) & (3)
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit

We will make a note of these and keep searching on.

Our last one is **BigTree 4.0.6**.
We have a feeling there's not going to be many results for this, so we are not going to include a version number directly in the search. Sometimes the web application is written as **"BigTree CMS"**, **"BigTreeCMS"**, **"BigTree-CMS"** or **"BigTree_CMS"**. So let's also remove the CMS part.

Code:

```
root@kali:~# searchsploit bigtree
...
```

```
root@kali:~# searchsploit bigtree
-----
Exploit Title | Path
-----
BigTree CMS 4.0 RC2 - Multiple Vulnerabilities | ./php/webapps/27431.txt
BigTree CMS 4.2.3 - Authenticated SQL Injection Vulnerabilities | ./php/webapps/37821.txt
root@kali:~#
```

There isn't anything which would be a "perfect" match.
There is a slim chance the 4.2.3 exploit might work, however theres been various subversion releases since then. Plus it requires authentication (which at this stage we do not have - nor even know of the login URL).

Summary
The highest change of success right now would be the PHP exploits:

- PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1), (2) & (3)
- PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability
- PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)
- PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit

Last edited by g0tmi1k; 09-12-2017 at 10:48 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply | Reply With Quote

Reply to Thread

« Previous Thread | Next Thread »

Posting Permissions

- | | |
|--------------------------|-------------------------|
| You may post new threads | BB code is On |
| You may post replies | Smilies are On |
| You may post attachments | [IMG] code is On |

You may edit your posts

[VIDEO] code is On
HTML code is Off

Forum Rules

-- Perfexion-Red ▾

| [Contact Us](#) | [Offensive Security Training](#) | [Archive](#) |

All times are GMT +1. The time now is 05:31 PM.
Powered by vBulletin® Version 4.2.4
Copyright © 2018 vBulletin Solutions, Inc. All rights reserved.
Offensive Security
Skin designed by: SevenSkins