Apache Security Log Analysis

Pre-Investigation: First look at an apache log.

Let's take a look at the logs we'll be working with. They're located at **Logs/apache_access.log**. Start by opening a terminal and typing the command below so that you're in the Logs directory:

```
cd Logs
```

Now let's look at the first ten logs.

```
head apache_access.log
```

Investigation #1: Strange ways of entering the site

Let's start by looking into how some users are accessing https://bluewaters.ncsa.illinois.edu/ The referer field in the Apache logs will get us that information. If there's a value, it means the user came from that page and was *referred* to the current one.

Run the following command to find out which site sends the most traffic to Blue Waters.

```
cat apache_access.log | awk '{print $11}' | sort | uniq -c | sort -n
```

What site is it? https://www.google.com/

Good information to have, but not really what we're looking for. You can see there is a lot of internal linking: people visit one page, and then click on a link on that page to visit a second page. Let's filter that out, and reverse our results so we see the ones with the fewest hits.

```
cat apache_access.log | awk '{print $11}' | grep -v bluewaters | sort | uniq
-c | sort -nr
```

Some of those look pretty suspicious. Choose a couple to drill down into, and look at what they accessed. Here's the command, just replace <> with what you want to search for:

```
cat apache_access.log | grep "<>"
```

It's odd that they're all accessing the same page. Which page is it? webinars/reusable-software

Definitely something we'd want to check out. Maybe that page was compromised in some way, and these other known bad sites are referring users to download malware from it.

Investigation #2: Access without using a referer

Sometimes logs won't have a referrer listed because the user came directly, either by bookmark, or by typing in the URL themselves. That's not necessarily a cause for concern, but it could be if they're doing something strange.

Let's use some more advanced awk to take a look at these. We'll only include logs that have "-" as their 11th field.

```
awk '{ if ($11 ~ /"-"/) {print $7} }' apache_access.log  | sort | uniq -c |
sort -nr
```

There's one URL that keeps showing up, with a lot of other garbage behind it. Which is it? (Only include what's before the question mark.) /assets/php/directory/list.php

We'll come back to that in a minute, but let's filter it out for now:

```
awk '{ if ($11 ~ /"-"/) {print $7} }' apache_access.log  | grep -v
"/assets/php/directory" | sort | uniq -c | sort -nr
```

What's all that stuff in /tmp? Let's look at that for a minute.

```
cat apache_access.log | grep "/tmp"
```

Well, now we know it's only one IP address accessing it. Which is it? 147.202.58.39
Let's find all access by that IP. Replace <> below with the IP address you found.

```
cat apache_access.log | grep "<>"
```

Looks like a lot of them received back 404 from the web server, meaning the page wasn't found, so someone was just poking around. As long as there weren't any code 200, meaning success, it's fine. Let's double-check by printing just the path and the server status for each out, and filtering out 404. Don't forget to replace <> with the IP address.

```
cat apache_access.log | grep "<>" | awk '{print $7, $9}' | grep -v 404
```

Oh no! What file did they find in inside the https://bluewaters.ncsa.illinois.edu/tmp/ directory?
/tmp/grants2017.xls

A user must have uploaded it and meant to delete it. It doesn't seem like this person poking around was supposed to have access!

Investigation #3: SQL Injection Attempts

Let's go back to those first weird logs found in the beginning of the last investigation. (Rerun the first command under investigation #3 to refresh your memory on what path they were hitting.) Let's take a look at some of those.

```
cat apache_access.log | grep "/assets/php/directory/list.php" | head
```

We have some interesting information in the last bit of each Apache line, and that's the user-agent - the string that the client's browser (or tool) identified itself as. Let's use awk to split on quotes instead of commas, and take a look:

```
cat apache_access.log | grep "/assets/php/directory/list.php" | head | awk -F
'"' '{print $6}'
```

Ah ha! What user-agent did you find? sqlmap/1.0.4.0#dev (http://sqlmap.org)

Whoever ran this is using a white-hat tool for nefarious purposes, but they forgot to change the user-agent, so the tool clearly gave itself away. sqlmap is a tool for investigating whether websites are vulnerable to SQL injection attacks.

Investigation #4: Strange user-agents, Part 1

That last investigation begs the question, what other strange user-agents might be showing up in the logs? Could they indicate anything malicious?

Let's use awk to split on quotes again, and look at all the user-agents accessing the site.

```
cat apache_access.log | awk -F '"' '{print $6}' | sort | uniq -c | sort -nr
```

Most of the commonly used ones look normal, but what's the one listed directly under the sqlmap entry? Struts-pwn (https://github.com/mazen160/struts-pwn

Does that word ring any bells? It's the bug that lead to the Equifax compromise. Someone's testing a tool against https://bluewaters.ncsa.illinois.edu/ to see if we're vulnerable. Let's see who it was.

```
cat apache_access.log | grep struts-pwn
```

Look like it wasn't just one individual. Finding a struts vulnerability on Blue Waters could be a gold mine. Unfortunately for them, NCSA patched weeks ago.

Investigation #5: Strange user-agents, Part 2

Go ahead and pick a couple of the other odd-looking user agents, and see what those individuals were up to. Which ones did you pick? _____ _____

If you didn't look at the one that includes the word "ping" in it, go ahead and grep for that one now.

```
cat apache_access.log | grep ping
```

That referer field looks like a command - and it is. This is someone attempting to use the shellshock vulnerability against the Blue Waters site. If the server is vulnerable, and the attack succeeds, the server would run the command listed, and phone home for the attacker to know they can start running whatever they want.

Luckily again, NCSA already patched as soon as the vulnerability was disclosed.

Investigation #6: Information-gathering

Investigation #2 unearthed a common pattern seen in the logs when someone attempts to gather information: they'll have a high number of 404 errors from the server, indicating that they tried to access a page that isn't available.

Run the command below to find the IP addresses with the highest number of 404 errors.

```
cat apache_access.log | awk '{print $1, $9}' | grep 404 | sort | uniq -c |
sort -n | tail
```

147.202.58.39 is our friend who was looking for files under https://bluewaters.ncsa.illinois.edu/tmp/ . What are the other two IP addresses? 89.163.222.7 34.98.100.25

Of those two addresses, what did the second one access? We can grep just for their IP address to find out. Don't forget to replace <> with the actual IP address.

```
cat apache_access.log | grep "<>"
```

They were definitely poking around. Did they find anything? Let's look at the second one, too (and don't forget to replace <> with the actual IP address):

```
cat apache_access.log | grep "<>"
```

Interesting, looks like they were accessing a lot of URLs that look like Jim's name. Let's view them with their statuses listed alongside.

```
cat apache_access.log | grep "<>" | awk '{print $7, $9}'
```

One of those directories sends back at 400 error, meaning bad request, rather than not found. Which is it? /~james-basney/ Any thoughts on why someone would do this, and what they gained?

Traditionally, many Linux servers allowed users to serve web content out of their home directories. These would served at https://<server_name>/~<username>/. Looks like someone was trying to find Jim's user name (and they succeeded). Now they can start trying to brute force his password via SSH on NCSA's servers.