CYBER SECURITY 2021

**P3: Log file analysis**

# Exercise:

I downloaded 3 files:

Graphical user interface, application

Description automatically generated

Figure 1. Files for inspection

When inspecting the fast.log.1 file, I notice that most the attacks are from IP 192.168.73.39. Also some attacks from 192.168.73.239 and 192.168.73.164

A picture containing text, outdoor

Description automatically generated

Figure 2. Attacks recorded in fast.log.1

And from file access.log.2 I find that there are 13 nikto attacks from 4 different Ips

A picture containing text

Description automatically generated

Figure 3. Nikto attacks

Actually I notice all the access to the server have (Nikto/2.1.6).

From the file fast.log.1, I found there are Web Application Attack, Generic Protocol Command Decode (seems like sql injection), Attempted Administrator Privilege Gain (bruteforce?), otential Corporate Privacy Violation

Snort is an open source IPS. Snort uses a series of rules to define malicious network activities and generates alert for users. Snort can also be deployed to stop these malicious packets as well.

**02/08/2019-12:53:12.516460  [\*\*] [1:2024364:3] ET SCAN Possible Nmap User-Agent Observed [\*\*] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.73.39:53395 -> 192.168.73.100:80**

This log means on 2/8/2019, there is an event thought to be an attack (Nmap scan) to the server at 192.168.73.100 from IP 192.168.73.39.

**02/08/2019-12:54:08.369551  [\*\*] [1:2019232:4] ET WEB\_SERVER Possible CVE-2014-6271 Attempt in Headers [\*\*] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.73.39:53417 -> 192.168.73.100:80**

This log means there was an attempt to gain privilege control from IP 192.168.73.39, which may refer to one of this event:

Text

Description automatically generated

Figure 4. Snort CVE-2014-627

**ET WEB\_SERVER Possible CVE-2014-6271 Attempt in HTTP Cookie:** The new vulnerability allows attackers to execute arbitrary commands formatting an environmental variable using a specific format. It affects Bash (the Bourne Again SHell), the default command shell for Linux and other UNIX flavors. The vulnerability is critical since it can be exposed on web servers that use mod\_cgi or code that calls the bash shell.

This is information from the page.

**02/08/2019-13:10:50.674921  [\*\*] [1:2016184:5] ET WEB\_SERVER ColdFusion administrator access [\*\*] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.73.39:55394 -> 192.168.73.100:80**

This log means there was an attempt trying to attack the web, which type is **WEB\_SERVER ColdFusion administrator access.** It allows an attacker to have unauthorized read and delete access to files on the target host.

Commands:

* grep: search for patterns in the given file(s). Some parameters:
  + -i: ignore case (upper or lower)
  + -v: display non-matching lines only (reverse)
  + -r -i ‘string’ ‘directory’: search for a string in a directory (and its subdirectories)
* sort: this tool help to quickly sort information. Some parameters:
  + -n: sorting by numerical value of the string (default is just the first character).
  + -r: sorting in reverse order.
  + -R: re-arrange output in a random order.
  + > ‘filename’: store the output to another file.
* uniq: this command looks for duplicate or unique lines.
  + -c: count the duplicates
  + -d: list the duplicated lines only (1 time)
  + -D: list all the duplicated lines
* awk ‘{print $1}’: This command is a filter to the output. It takes only the first section ($1), each section is separated by the blank space ‘ ‘. This can be useful in pipe commands.
* cut: allows users to cut parts of lines from given file(s) or piped data and print the output to standard terminal.
  + -f: specify a field or a set of fields
  + -b: specify a byte or a set of bytes
  + -c: specify a character or a set of characters.
  + -d: (delimiter) specify a delimiter instead of the default “TAB”

The command **grep access.log|awk '{print $1}'|sort|uniq**: filter out the first section only from the file access.log (the IP address), sorts and remove the duplicated results. But when I use **grep** it does not run, so I use **cat** instead:

Text

Description automatically generated

Figure 5. First p iped commands

The command **cut -d '"' -f 3 /var/log/apache/access.log | cut -d' ' -f2 | sort | uniq -c | sort -rg**: First, cut out the third field only from the access.log file (fields are separated by “); then cut out the second field from that (now we have the PATHs of file). The results are alphabetically sorted, then count for unique lines. Last command is to reversely sort the list based on generic numerical value of the string.

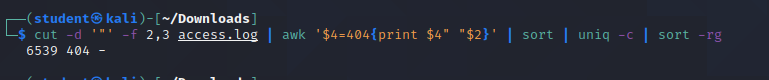
Text

Description automatically generated

Figure 6. Second piped command

Try also these commands:  
1- Most Common 404s (Page Not Found):

**cut -d '"' -f 2,3 /var/log/apache/access.log | awk '$4=404{print $4" "$2}' | sort | uniq -c | sort -rg**



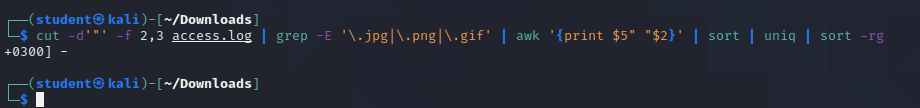
2 - Count requests by HTTP code:

**cut -d'"' -f3 /var/log/apache/access.log | cut -d ' ' -f 2 | sort | uniq -c | sort -rg**

(done above)

3 - Largest Images

**cut -d'"' -f 2,3 /var/log/apache/access.log | grep -E '\.jpg|\.png|\.gif' | awk '{print $5" "$2}' | sort | uniq | sort -rg**



4 - Filter Your IPs Requests (run command and access yoyw local web server with your browser)

**tail -f /var/log/apache/access.log | grep <your IP>**

(cannot do that, same as command 5)