CS 3710 Introduction to Cybersecurity
Term: Spring 2023

Lab Exercise 7 – Binary Analysis, Firewall, and Intrusion Detection

Due Date: April 14, 2023 11:59pm
Points Possible: 7

1. Overview

This lab exercise will provide some hands-on experience with binary analysis, firewall configuration, and intrusion detection.

2. Resources required

This exercise requires Kali Linux VM running in the Virginia Cyber Range.

3. Initial Setup

From your Virginia Cyber Range course, select the **Ubuntu with Snort and Other Tools** environment. Click "start" to start your environment and "join" to get to your Linux desktop login. This environment requires authentication. Log in using these credentials:

Username: **`student`**
Password: **`student`**

Once you are logged in, click the Terminal Emulator in the bottom menu to open the command line.

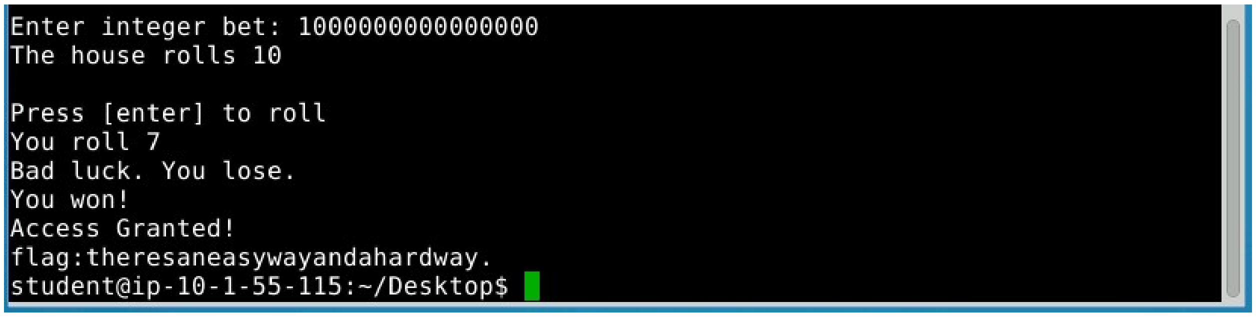
4. Tasks

Task 1: Binary Analysis

Run the file **game** and win the game by capturing the flag. You can use the various static and dynamic tools and a little fuzzing to determine the best way to trick the game and win. The file is already located on the Desktop in Cyber Range but you may need to install the analysis tools you want to use. If you want to use your own system you can download the game file here:

bit.ly/3Cc5QEo

Question 1: **Provide a screenshot of what you entered to win the game (make sure it shows the flag). (.5 point)**



```
Enter integer bet: 1000000000000000
The house rolls 10

Press [enter] to roll
You roll 7
Bad luck. You lose.
You won!
Access Granted!
flag:theresaneasywayandahardway.
student@ip-10-1-55-115:~/Desktop$
```

Task 2: Firewall Configuration

[Note: be very careful with firewall rule configuration changes on your Cyber Range virtual machine. If you set the rules improperly you could break your network connection to the range VM. Fortunately, this can almost always be fixed by restarting your VM. If that happens, go to the Virginia Cyber Range page and select the "Stop Exercise" button for this lab, then restart the exercise and re-join.]

Use the following command to set the host-based firewall on your Linux system to a default policy that we have specified:

```
$ sudo /etc/default_firewall.sh
```

[When using ***sudo*** you may need to enter your student password: ***student***]

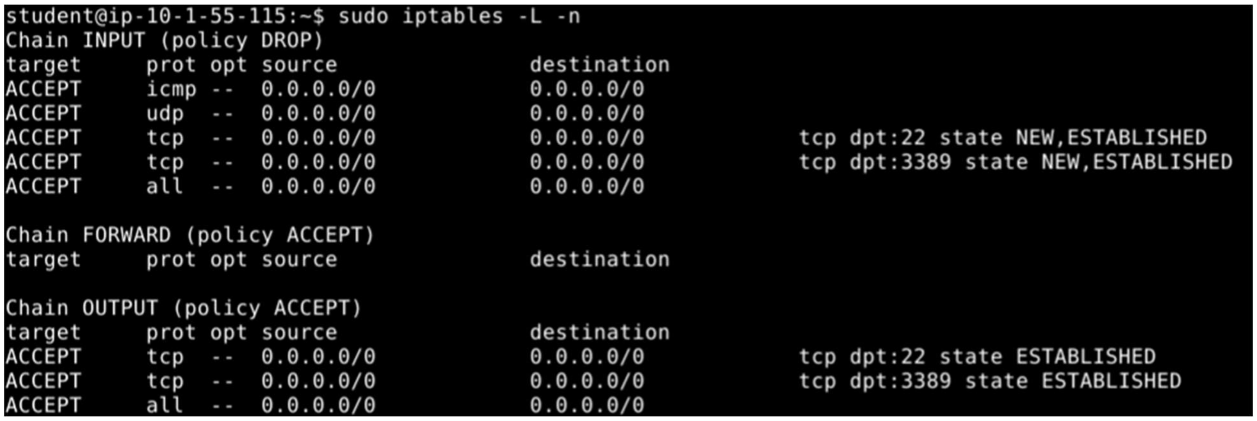
Linux host-based firewalls are configured using the **iptables** command. There is a pretty good (and short) tutorial at http://fideloper.com/iptables-tutorial. To review the firewall rules set by the default policy, use the following command (you may want to use the mouse to drag the terminal screen wider first to read the output more clearly with no linebreaks):

```
$ sudo iptables -L -n
```

Simple packet filtering firewalls usually have a default policy to DROP (deny) packets and only to accept traffic that meets specific criteria. When a packet arrives on a host, the firewall tries to match firewall rules starting with the first rule in the chain. The firewall will apply the first rule that matches and the default rule is applied last, so if there is a rule that "ACCEPTs" a packet before the default DROP, the packet will be accepted. In general, if a specific input or output IP address, port, or protocol is not specified, the rule applies to 'any' IP address, port, or protocol.

Review the default firewall configuration (**$ sudo iptables -L -n**) and answer questions 1 – 3.

Question 2: **What is the default policy on the INPUT, FORWARD, and OUTPUT chains in the default firewall configuration and what does this mean for each? (.5 point)**



```
student@ip-10-1-55-115:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22 state NEW,ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:3389 state NEW,ESTABLISHED
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:22 state ESTABLISHED
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            tcp dpt:3389 state ESTABLISHED
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
```

The default policy for the INPUT chain is DROP. This means that packets that don't conform to the INPUT chain are dropped/discarded. The default policy for the FORWARD and OUTPUT

chains is ACCEPT. This means that packets that FORWARD and OUTPUT packets that don't conform to the chains in their respective sections are accepted even if they don't conform to the given chain rules.

Question 3: **What specific firewall rules are in place on the INPUT chain? Specify protocols and ports for which packets are allowed by the rules provided, and under what conditions those packets are allowed. (Hint: there are 5 rules) (.5 point)**

We have five rules for the INPUT chain. They are as follows:

1. **We accept incoming packets that are on the ICMP protocol**
2. **We accept incoming packets that use the udp protocol**
3. **We accept incoming packets that use the TCP protocol on port 22 with either NEW or ESTABLISHED states**
4. **We accept incoming packets that use the TCP protocol on port 3389 with either NEW or ESTABLISHED states**
5. **We accept all other traffic**

Question 4: **You notice a big problem with the firewall rules on the INPUT chain. What is it? (.5 point)**

Although we have placed chain RULES on incoming traffic, at the end we ACCEPT all packets anyway. Disregarding the DROP default policy.

Question 5: **What firewall rules are in place on the OUTPUT chain? Specify protocols and ports for which packets are allowed by the rules provided, and under what conditions those packets are allowed. (.5 point)**

We have three rules for the OUTPUT chain. They are as follows:

1. **We accept outgoing TCP packets on port 22**
2. **We accept outgoing TCP packets on port 3389**
3. **Finally, we accept all outgoing traffic**

We will use two shell scripts to modify the firewall configuration. A script called '/etc/extingui.sh' will clear all firewall rules and set the default policy on the INPUT, OUTPUT, and FORWARD chains to ALLOW all traffic in and out of your server. Execute this script as follows.

```
$ sudo /etc/extingui.sh
```

Perform the following command again to see that the firewall rules are cleared:

```
$ sudo iptables -L -n
```

Once the firewall rules are cleared, you will modify the script '/home/student/lab2/firewall.sh' to add firewall configuration commands. This file is not a blank file, it already has a firewall rules template in it. Use the text editor of your choice to edit this script (one option is "mousepad".)

```
$ mousepad /home/student/lab2/firewall.sh
```

Iptables commands are of the following form:

```
iptables [command-type] [pattern-match options] -j [action]
```

Where `[command-type]` specifies whether the rule will be added or deleted on a specified chain, `[pattern-match-options]` specifies the port, interface, address, etc. to match, and `[action]` specifies what action to take if the packet matches the pattern (DROP, REJECT, ACCEPT, LOG).

In our simple packet filtering firewall, all of our rules will be added to the INPUT or OUTPUT chains and our actions will either be ACCEPT or DROP; so in this exercise, all of your rules will be of the form:

```
iptables -A INPUT  [pattern-match options]  -j [ACCEPT or DROP]
```

Pattern match options that you will use include:

-s	source IP address or address range (can use CIDR addressing)
-d	destination IP address or address range
-p	transport layer protocol (tcp, udp, or icmp)
-m	match a specific property (such as 'state')
--dport	destination port number (must be used with a protocol specified by the -p option)
--sport	source port number (must be used with a protocol specified by the -p option)
--state	connection state (NEW, ESTABLISHED, etc.)

An example rule using the above options is here:

```
# Allow inbound packets to TCP port 20 from subnet 192.168.1.0/24
iptables -A INPUT -s 192.168.1.0/24 -p tcp --dport 20 -j ACCEPT
```

Add rules to your /home/student/lab2/firewall.sh script that will allow outbound connection attempts on port 80 and the return traffic. The rules must perform stateful inspection and include the interface, protocol, and ports. Be very specific with your rules and include state, don't just allow all.

Once you have edited and saved the firewall.sh file, apply at the command line as follows:

```
$ sudo /home/student/lab2/firewall.sh
```

Perform the following command again to see that the new firewall rules are added:

```
$ sudo iptables -L -n
```

Question 6: **List the rule(s) that you added to the firewall.sh to allow outbound HTTP requests (port 80) and responses. (1 point)**

INPUT chain rule:

iptables -A INPUT -p tcp –dport 80 -m state –state ESTABLISHED

iptables -A OUTPUT -p tcp –sport 80 -m state –state NEW,ESTABLISHED

The first allows incoming TCP traffic via established TCP channels on port 80. This ensures that external machines can't establish new TCP connections with our machine. The second rule allows outgoing TCP traffic via new and established TCP channels.

Task 2: Intrusion Detection

Your Virginia Cyber Range virtual machine has Snort software installed for intrusion detection. Instead of observing traffic from a network interface, we will use Snort to process packet capture files (.pcap files) from previously captured traffic.

Here is a great Snort reference from the Snort creator:
https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm

Before we run Snort against captured packets, we'll take a look at some snort rules (signatures) in the /etc/snort/rules directory. To do this, open a terminal window and change to the appropriate directory as follows.

```
$ cd /etc/snort/rules
$ ls                    ← this will list all the rule files
```

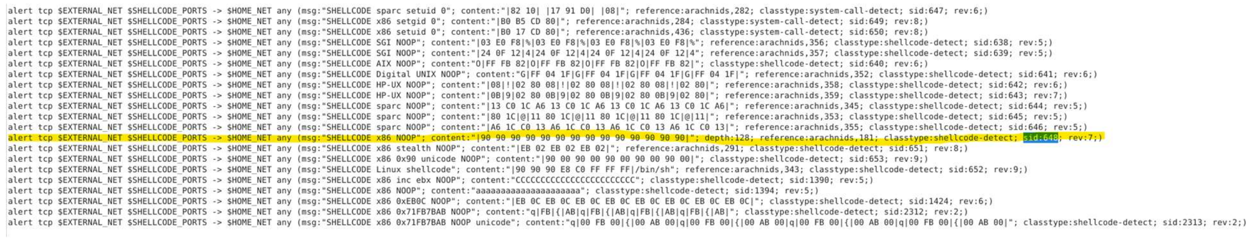
Examine the file **`shellcode.rules`** using the text editor of your choice (your Linux VM includes *vi* and *nano*, as well as a GUI text editor called *mousepad* as shown in the command below. You could also use the *cat* or *more* command).

```
$ mousepad shellcode.rules &
```

Each rule has a unique Snort ID number (sid), which is included in the signature. In *shellcode.rules*, find **sid:648** amongst the rules in that file and answer the following questions.

Question 7: **What is the specific signature (content) that sid:648 tries to match? (.5 point)**

sid:648 tries to match content that features a sequence of nops. In this case a sequence of 14 nop codes within the packet.



```
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE sparc setuid 0"; content:"|82 10| |17 91 D0| |08|"; reference:arachnids,282; classtype:system-call-detect; sid:647; rev:6;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 setgid 0"; content:"|B0 B5 CD 80|"; reference:arachnids,284; classtype:system-call-detect; sid:649; rev:8;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 setuid 0"; content:"|B0 17 CD 80|"; reference:arachnids,436; classtype:system-call-detect; sid:650; rev:8;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE SGI NOOP"; content:"|03 E0 F8|%|03 E0 F8|%|03 E0 F8|%|03 E0 F8|%"; reference:arachnids,356; classtype:shellcode-detect; sid:638; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE SGI NOOP"; content:"|24 0F 12|4|24 0F 12|4|24 0F 12|4|24 0F 12|4"; reference:arachnids,357; classtype:shellcode-detect; sid:639; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE AIX NOOP"; content:"O|FF FB 82|O|FF FB 82|O|FF FB 82|O|FF FB 82|"; classtype:shellcode-detect; sid:640; rev:6;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE Digital UNIX NOOP"; content:"G|FF 04 1F|G|FF 04 1F|G|FF 04 1F|G|FF 04 1F|"; reference:arachnids,352; classtype:shellcode-detect; sid:641; rev:6;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE HP-UX NOOP"; content:"|08|!|02 80 08|!|02 80 08|!|02 80 08|!|02 80|"; reference:arachnids,358; classtype:shellcode-detect; sid:642; rev:6;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE HP-UX NOOP"; content:"|0B|9|02 80 0B|9|02 80 0B|9|02 80 0B|9|02 80|"; reference:arachnids,359; classtype:shellcode-detect; sid:643; rev:7;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE sparc NOOP"; content:"|13 C0 1C A6 13 C0 1C A6 13 C0 1C A6 13 C0 1C A6|"; reference:arachnids,345; classtype:shellcode-detect; sid:644; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE sparc NOOP"; content:"|80 1C|@|11 80 1C|@|11 80 1C|@|11 80 1C|@|11|"; reference:arachnids,353; classtype:shellcode-detect; sid:645; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE sparc NOOP"; content:"|A6 1C C0 13 A6 1C C0 13 A6 1C C0 13 A6 1C C0 13|"; reference:arachnids,355; classtype:shellcode-detect; sid:646; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth:128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:7;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 stealth NOOP"; content:"|EB 02 EB 02 EB 02|"; reference:arachnids,291; classtype:shellcode-detect; sid:651; rev:8;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 0x90 unicode NOOP"; content:"|90 00 90 00 90 00 90 00 90 00|"; classtype:shellcode-detect; sid:653; rev:9;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE Linux shellcode"; content:"|90 90 90 E8 C0 FF FF FF|/bin/sh"; reference:arachnids,343; classtype:shellcode-detect; sid:652; rev:9;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 inc ebx NOOP"; content:"CCCCCCCCCCCCCCCCCCCCCCCC"; classtype:shellcode-detect; sid:1390; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 NOOP"; content:"aaaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-detect; sid:1394; rev:5;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 0xEB0C NOOP"; content:"|EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C|"; classtype:shellcode-detect; sid:1424; rev:6;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 0x71FB7BAB NOOP"; content:"q|FB|{|AB|q|FB|{|AB|q|FB|{|AB|q|FB|{|AB|"; classtype:shellcode-detect; sid:2312; rev:2;)
alert tcp $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86 0x71FB7BAB NOOP unicode"; content:"q|00 FB 00|{|00 AB 00|q|00 FB 00|{|00 AB 00|q|00 FB 00|{|00 AB 00|q|00 FB 00|{|00 AB 00|"; classtype:shellcode-detect; sid:2313; rev:2;)
```

***Question 8:* What action is Snort supposed to take if the signature contained in sid:648 is matched? (.5 point)**

> **If it finds a signature contained in sid:648, it will send a alert message to the system with the content "SHELLCODE x86 NOOP".**

Change directories to **/home/student/lab2** and run **snort** against the packet capture file called **theft.pcap** in that directory as shown here.

```
$ sudo snort -c /etc/snort/snort.conf -r theft.pcap
```

[When using ***sudo*** you may need to enter your student password: ***student***]

You will see Snort processing on your screen, let it complete and return on the command prompt $.

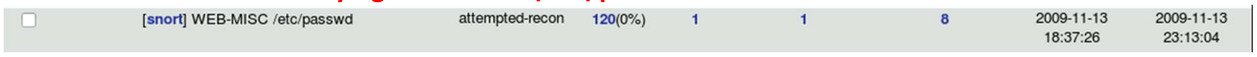
When Snort finishes processing, open a web browser on your Cyber Range virtual machine (on the menu bar at the bottom of the screen) It will automatically open the following URL: http://localhost/base.

BASE is the Basic Analysis and Security Engine, which is installed on your virtual machine along with Snort. It allows you to view Snort alerts in a nice graphical format. Log in to BASE using the username: **john** and the password: **secret3**. The homepage should show the results of the scan we just processed using Snort, with a little over 27,000 Total Number of Alerts. [If you aren't seeing "Total Number of Alerts: 27081, It takes a few minutes for the back-end alert processing to complete, so you might have to refresh the page a few times.] NOTE: If you run the command with theft.pcap more than once you will have double or triple the alerts, **so only run it once**.

Review the alerts (you might have to do some filtering) and answer the following questions.

***Question 9:* An attacker was trying to steal a specific, and very sensitive, file from the target system. What file was she after? (Hint: Click on the Total Number of Alerts, then click on Unique Alerts to filter the information) (.5 point)**

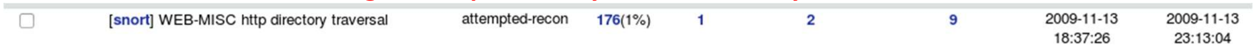
> **The attacker was trying to steal the /etc/passwd file.**



☐	[snort] WEB-MISC /etc/passwd	attempted-recon	120(0%)	1	1	8	2009-11-13 18:37:26	2009-11-13 23:13:04

***Question 10:* What is the technique that the attacker was trying to use to steal the file? (1 point)**

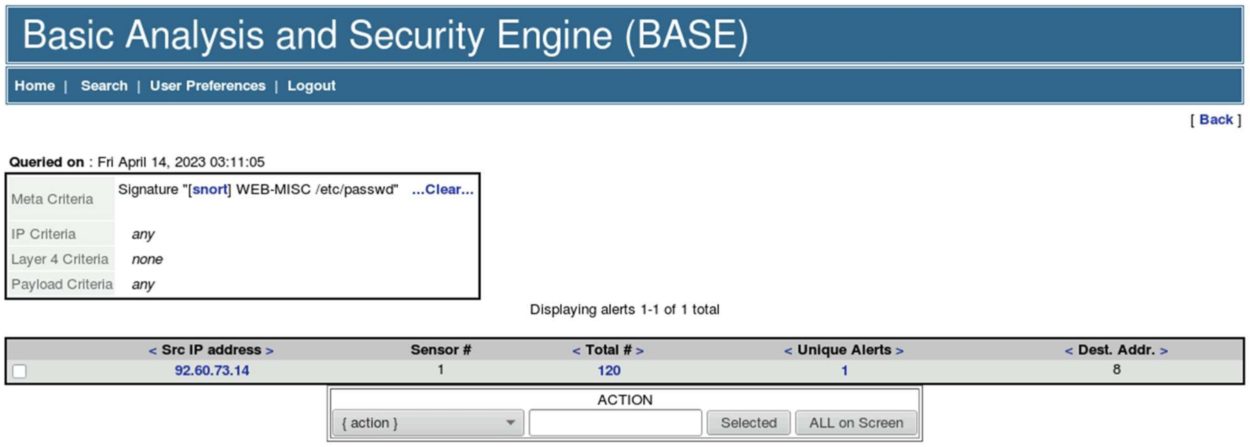
The attacker was using an "http directory traversal" to try and steal the file.



☐	[snort] WEB-MISC http directory traversal	attempted-recon	176(1%)	1	2	9	2009-11-13 18:37:26	2009-11-13 23:13:04

***Question 11:* What is the source IP address of the attacker that is trying to steal the file from the system? (1 point)**

The source IP address of the attacker trying to steal the /passwd file from the system is 92.60.73.14


Basic Analysis and Security Engine (BASE)
Home | Search | User Preferences | Logout
[Back]

Queried on : Fri April 14, 2023 03:11:05

Meta Criteria	Signature "[snort] WEB-MISC /etc/passwd" ...Clear...
IP Criteria	any
Layer 4 Criteria	none
Payload Criteria	any

Displaying alerts 1-1 of 1 total

	< Src IP address >	Sensor #	< Total # >	< Unique Alerts >	< Dest. Addr. >
☐	92.60.73.14	1	120	1	8

ACTION
{ action }
Selected
ALL on Screen

By submitting this assignment you are digitally signing the honor code, "I pledge that I have neither given nor received help on this assignment".

END OF EXERCISE

References

- iptables man page: https://linux.die.net/man/8/iptables
- iptables tutorial: https://fideloper.com/iptables-tutorial
- Snort: https://www.snort.org/
- Writing Snort Rules: https://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm
- BASE: https://lwn.net/Articles/112548/