Department of Electronic & Computer Engineering

Intrusion Detection Systems Assignment

ET4028 – Host and Network Security

Student Name: Dylan Coffey

Student ID: 18251382

Lecturer: Reiner Dojen

Date: 15/04/2022

Environment Setup:

This assignment was completed using two virtual machines on VMware. The victim machine uses Ubuntu 20.04 LTS and the Snort IDS, while the attacker machine uses Kali Linux 64-bit. The IP addresses for each machine can be seen below:

Victim: *192.168.255.128*

Attacker: *192.168.255.129*

Within the *"snort.conf"* file, the default include rules were all removed to reduce the number of entries that will show in the logs. The network variables in the file were also set to have the following addresses:

Line no: 51 *ipvar HOME_NET 192.168.255.128*

Line no: 54 *ipvar EXTERNAL_NET any*

Attack #1 – Null Port Scan

Attack Description:

The first attack was a null port scan using Nmap. This is when a packet with no flags set is sent to TCP ports. If a port is open, it won't send a response as it cannot handle the request of the packet. The target will only send a response if the port is closed. This allows for the discovery of TCP ports that are listening. The following command was used for the attack:

nmap -sN 192.168.255.128

Component	Description
nmap	Tool – nmap (network mapper and port scanner)
-sN	Scan Type – null port scan
192.168.255.128	Target Specification – IP address of the victim

Attack Location (Port 80):

Alert File: *alert_port_scan* Line no: 25-29

Log File: *snort.log.1650044857* Line no: 5

Rule Location:

Config File: *snort.conf* Line no: 573

Rule Description:

alert tcp $EXTERNAL_NET any -> $HOME_NET 1:100 (msg:"ALERT - Null Port Scan"; flags:0; classtype:attempted-recon; sid:1000001; rev:1;)

Component	**Description**
alert	Rule Action – an alert will generate when condition is met
tcp	Traffic Type – TCP (Transmission Control Protocol)
$EXTERNAL_NET	Source IP Address – any address
any	Source Port – any port
->	Direction – from source to destination
$HOME_NET	Destination IP Address – 192.168.255.128
1:100	Destination Port – ports from 1 to 100 (to reduce alert entries)
msg:"ALERT - Null Port Scan";	Message – included with the alert for human readability, saying it is for a null port scan
flags:0;	Flags – will look for packets with 0 flags set
classtype:attempted-recon;	Classification Type – helps with the organisation of rules, classifying it as an *"*attempted information leak*"*
sid:1000001;	Snort Rule ID – unique rule identifier, rule IDs < 1,000,000 are reserved
rev:1;	Revision Number – unique rule revision number, allows for easier rule maintenance

Attack #2 – ICMP Flood DoS

Attack Description:

The second attack was an ICMP flood DoS (denial of service) using Hping3. This is when an attacker is attempting to shut down a machine by overwhelming it with ICMP echo request packets. A more effective form of attack would be to use DDoS (distributed denial of service), as it coordinates many machines to attack the target at the same time. However, in this case, a single machine was used, which sends a custom packet filled with the data string of: *"This is an ICMP flood DoS attack!"*. The following command was used for the attack:

hping3 -d 34 -E data.txt --fast --icmp 192.168.255.128

Component	Description
hping3	Tool – hping3 (sends packets to network hosts)
-d 34	Data Size – size of the content that is filling the packets data
-E data.txt	File Name – uses the content of the specified file to fill the packets data
--fast	Speed – sends 10 packets every second
--icmp	Protocol – will send ICMP echo requests
192.168.255.128	Hostname – IP address of the victim

Attack Location (First Packet):

Alert File: *alert_dos* Line no: 1-5

Log File: *snort.log.1650044890* Line no: 1

Rule Location:

Config File: *snort.conf* Line no: 574

Rule Description:

alert icmp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"ALERT - ICMP Flood DoS"; pcre:"/icmp\s+flood/i"; itype:8; detection_filter: track by_src, count 50, seconds 10; classtype:denial-of-service; sid:1000002; rev:1;)

Component		Description
alert		Rule Action – an alert will generate when condition is met
icmp		Traffic Type – ICMP (Internet Control Message Protocol)
$EXTERNAL_NET		Source IP Address – any address
any		Source Port – any port
->		Direction – from source to destination
$HOME_NET		Destination IP Address – 192.168.255.128
80		Destination Port – port 80 (HTTP)
msg:"ALERT - ICMP Flood DoS";		Message – included with an alert for human readability, saying it is for an ICMP flood DoS attack
pcre:"/icmp\s+flood/i";		Perl Compatible Regular Expressions – will look for the content between the forward slashes, ignoring case sensitivity using “i”, and allowing for any number of whitespaces between the content using “\s+”
itype:8;		ICMP Type – will look for packets with type 8 (echo request) in their header
detection_filter:	track by_src,	Address Option – will track the rate by each unique source IP address
	count 50,	Count Option – the maximum number of rule matches allowed is 50 (in the seconds specified)
	seconds 10;	Seconds Option – the period for the count is 10 seconds
classtype:denial-of-service;		Classification Type – helps with the organisation of rules, classifying it as a “detection of a denial of service attack”
sid:1000002;		Snort Rule ID – unique rule identifier, rule IDs < 1,000,000 are reserved
rev:1;		Revision Number – unique rule revision number, allows for easier rule maintenance

Attack #3 – SSH Brute Force

Attack Description:

The final attack was a SSH brute force using Hydra. This is when an attacker uses a dictionary of logins and passwords to attempt to gain access to a server through SSH. Through trial and error, every combination of the dictionary is attempted, and if connection is permitted, the correct credentials are returned. In this case, a custom dictionary was used with the login cracking tool Hydra. The following command was used for the attack:

hydra -L logins.txt -P passwords.txt ssh://192.168.255.128 -t 4

Component	Description
hydra	Tool – hydra (a network login cracker which supports SSH)
-L logins.txt	Logins – uses the list of logins in the file specified
-P passwords.txt	Passwords – uses the list of passwords in the file specified
ssh://192.168.255.128	Service/Target – connects to the SSH service of the victim IP address
-t 4	Tasks – the number of tasks that attempt connection in parallel is 4

Attack Location (First Login Attempt):

Alert File: *alert_brute_force* Line no: 1-6

Log File: *snort.log.1650044977* Line no: 1

Rule Location:

Config File: *snort.conf* Line no: 575

Rule Description:

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"ALERT - SSH Brute Force"; content:"SSH-"; nocase; depth:4; flow:to_server; detection_filter: track by_src, count 5, seconds 30; classtype:unsuccessful-user; sid:1000003; rev:1;)

Component		Description
alert		Rule Action – an alert will generate when condition is met
tcp		Traffic Type – TCP (Transmission Control Protocol)
$EXTERNAL_NET		Source IP Address – any address
any		Source Port – any port
->		Direction – from source to destination
$HOME_NET		Destination IP Address – 192.168.255.128
22		Destination Port – port 22 (SSH)
msg:"ALERT - SSH Brute Force";		Message – included with an alert for human readability, saying it is for a SSH brute force attack
content:"SSH-";		Content – will look for the content “SSH-” in the packet
nocase;		No Case Sensitivity – won’t check case sensitivity for the content
depth:4;		Byte Depth – the depth of bytes to check in each packet
flow:to_server;		Flow – will look to see if traffic is flowing from client to server
detection_filter:	track by_src,	Address Option – will track the rate by each unique source IP address
	count 5,	Count Option – the maximum number of rule matches allowed is 5 (in the seconds specified)
	seconds 30;	Seconds Option – the period for the count is 30 seconds
classtype:unsuccessful-user;		Classification Type – helps with the organisation of rules, classifying it as an “unsuccessful user privilege gain”
sid:1000003;		Snort Rule ID – unique rule identifier, rule IDs < 1,000,000 are reserved
rev:1;		Revision Number – unique rule revision number, allows for easier rule maintenance