Ryan Vacca

Julia Vrooman

Mathew Moltzau

**Report Part 2**

*Bash Scripting*

**Introduction**: The bash scripting program developed on Kali Linux is designed to scan every single IP-Address and all associated port numbers on a local area network. After the scan, the program needs to determine when a new IP-Address has been added to the network or a new port has been open for data transfer. This alert needs to be printed to a log file displaying the IP-Address and port number recently opened / connected. A scan of the LAN needs to occur every 5 minutes to check for new activity.

**Implementation**: The following will describe the code layout to better understand how this task is being accomplished.

1. **Files** – The uses 3 prepopulated files. Oldfile.txt and newfile.txt will be the files in which the program performs the comparison against.
   1. **oldfile.txt** : This will hold the previous scan values, IP’s + Ports
   2. **Newfile.txt** : This will hold the next scan values, IP’s + Ports
   3. **logfile.txt** : This will hold the alerts for new IP’s / Ports
2. **Nmap** – The program makes use of the nmap tool in order to query the network scanning for IP’s connected a Ports open. We use the command in two major ways in the program.
   1. **IP-Address Scan** : First scan for all IP address connected to the network. We use the -n option for optimization to turn off reverse name resolution. We use the “-sn” option to not scan ports just yet. Since at this stage we only want IP address. then the greppable output is piped into an “awk” command for formatting so that we only get the IP address as output. Mapfile is then used to populate each IP address into an array.
   2. **Port Scan** : Next scan using the IP’s on the LAN as input into nmap to produce open ports for those IP address’s.
3. **Comparison –** After the appropriate fields have been updated we are now ready to make the comparison between the two files. We use the Unix command “comm” to first compare the two files to check for changes. This return an exit case in which we can decide if the files were different. If they are different then we know a new IP / Port has surfaced. We then use “comm” with the options “-23” to only print the difference in the newfile.txt. This allows an IP-Address to fall off the network along with a closing of a port and not have the program register as if a new connection / port opening has occurred since the files were different.
4. **Output – The program outputs messages to STDOUT and the log.txtfile.**
   1. **logfile.txt :** The program outputs to the logfile.txt a status of each comparison scan. The output is either “No New IP address or Open Port” or “New IP address or Open Port” followed by the newly discovered Item.
   2. **Stdout :** Status updates will be sent to the console.
      1. Prior to entering a new scan.
      2. After the scan has completed.
      3. How long the program is going to sleep for.
5. **Delay –** A five-minute delay is imposed per project requirements and the program makes use of the bash SECONDS feature to get the time from when the script started in seconds. Since the nmap scan takes time in itself, we subtract the scan time from the 5-minute wait time to ensure a scan starts at exactly the 5-minute interval.

**Test Case**

The following test case models the same environment explained in the HW instructions. We set up the LAN on 192.168.10.\*/24. The bash script will run on Kali VM1. At first, only Kali VM1 and Windows VM are running.

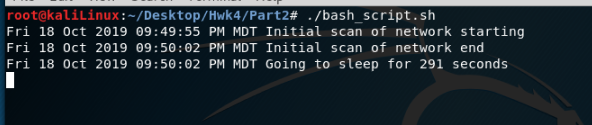
After 10 minutes, run a web application on port 8008, using SimpleHTTPServer of Python.

This runs a Web application on Kali VM2 and opens up port 8008.

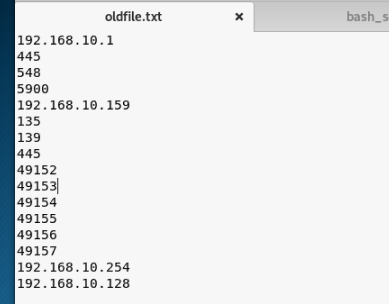
During these activities, your bash script is running every 5 minutes and logging the changes.

**TIME = 5 minutes into program**

**Step 1:** The bash script will run on Kali VM1. At first, only Kali VM1 and Windows VM are running.

****

*Shows console after initial scan entering the wait*

****

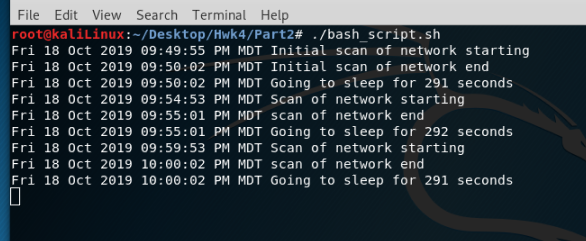
*Shows file with captured IP / Ports*

*\*Windows Server VM is 192.168.10.159*

*\*Kali Linux running the program is 192.168.10.128*

**TIME = 10 minutes into program**

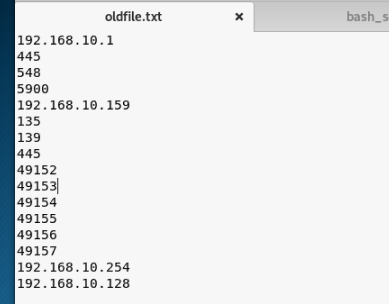
**Step 2:** Then, after 10 minutes, power on Kali VM 2 on the same VMnet0 network. This will not register until the third scan at 15 minutes.

**

*Shows console after the second comparison.*

****

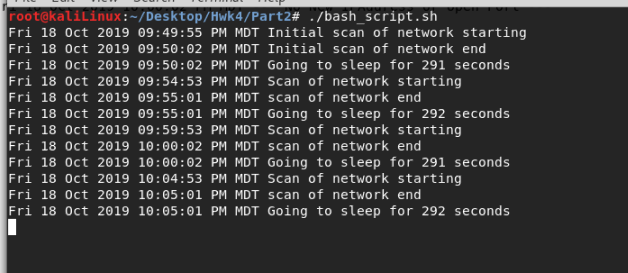
*Shows the logfile after the 3rd scan “2nd comparison scan”*

**

*Shows file unchanged as expected following second comparison since nothing has been added to the network at the ten minute point. The newly added 2nd Kali VM will not register until the 3rd comparison below.*

**TIME = 15 minutes into program**

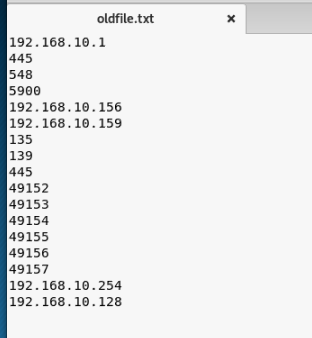
**Step: Do Nothing**

**

*Shows console after 3rd comparison “Adding 2nd Kali VM to network”*

**

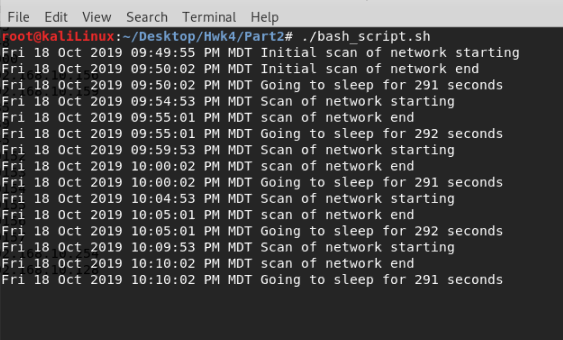
*Shows logfile registering 2nd Kali VM as a new IP-Address on the network and printing the appropriate message in the logfile along with the address number. 2nd Kali VM IP-Address = 192.168.10.156*

**

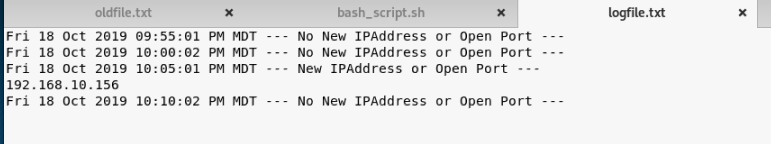
*Shows comparison file registering the 2nd Kali VM.*

**TIME = 20 minutes into program**

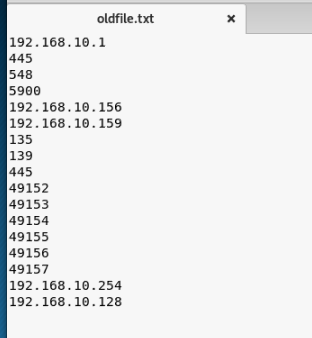
Step 3: After 10 minutes, run a web application on port 8008, using SimpleHTTPServer of Python.This runs a Web application on Kali VM2 and opens up port 8008.

**

*Shows console after 4th comparison, we now power on the python server in the second Kali VM.*

**

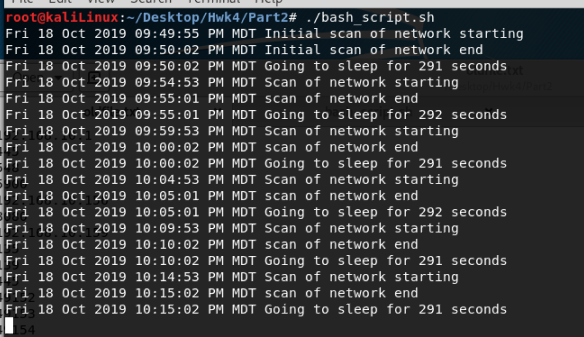
*Show logfile registering no change since nothing was added. Wont register python server till next comparison.*

**

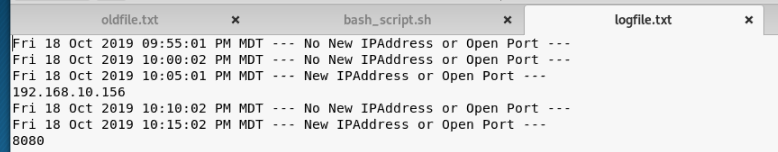
*File for comparison, no change.*

**TIME = 25 minutes into program**

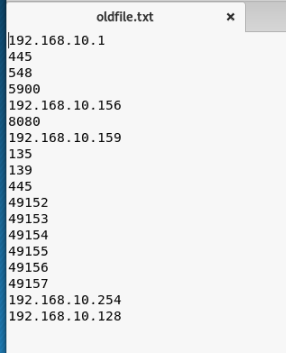
**Step: Do Nothing**

**

*Show console after 5th comparison.*

**

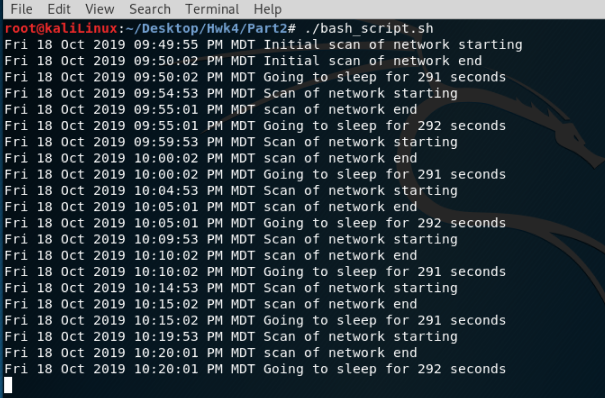
*Show Logfile now registering the python server through port 8080*

**

*Show comparison file*

**TIME = 30 minutes into program**

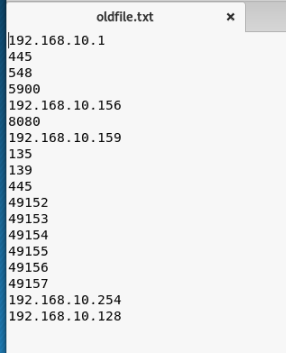
**Step: Do Nothing**

**

*Show logfile after the 6th comparison.*

**

*Show logfile after registering the newly opened port last comparison now registering nothing new.*

**

*Show file for comparison no change from last run.*