Research Article

Journal of Medical - Clinical Research & Reviews

A Comprehensive Analysis of Security Tools for Network Forensics

Christopher Lopez-Araiza and Ebru Celikel Cankaya

University of Texas at Dallas Department of Computer Science, Richardson, TX, USA.

*Correspondence:
Ebru Celikel Cankaya, University of Texas at Dallas Department of Computer Science, Richardson, TX, USA, E-mail: ebru.cankaya@utdallas.edu.

Received: 21 October 2017; Accepted: 23 November 2017

Citation: Christopher Lopez-Araiza, Ebru Celikel Cankaya. A Comprehensive Analysis of Security Tools for Network Forensics. J Med - Clin Res & Rev. 2017; 1(3): 1-9.

ABSTRACT

In an effort to establish a standard for responsive networking systems, we provide a survey of available tools and their applications for network forensics, as well as discuss the accessibility of these solutions to implement. Our paper investigates four network security tools in detail: Fail2ban, Netdata, Nmap, and HoneyDrive3 to test run on experimental setup. We compare these tools w.r.t. seven fundamental forensics criteria as logging, automated threat response, active monitoring, attack prevention capability, malicious activity detection, malicious activity notification, and security auditing. Experimental results are compared for further analysis. We rank results based on degree of coverage for the full set of seven forensics criteria. We also emphasize how utilizing relevant solutions could have aided in mitigating past threats.

Keywords
Security Tools, Network Forensics.

Introduction
As new technologies become integrated with our daily lives, we have seen a growing desire for data. Smart Devices like cellphones, televisions, and appliances provide another source of input and output into our day. Data collection, theft, and utilization has now become common. Businesses, as well as individuals, have begun to realize data is currency to the right customer. There are professionals on all sides of the front, yet the problem of attempting to preempt the unexpected remains. New forensics tools and improvements are on the rise, yet standards appear slow to follow leaving the average user vulnerable. Many network forensics tools and approaches attempt to solve the problem in retrospect, rather than focusing efforts into active response and monitoring. Common preemptive measures seek to collect data and log actions for later review after an incident has occurred. This approach leaves forensic responders and security specialists behind from the start. Similar tools available today can provide insight to these same incidents as they occur, and provide a foundation to create a standard for quicker threat response. This paper surveys some available tools and their applications for network forensics, while advocating for a more responsive standard of network security. Through this paper we seek to draw attention to the importance of a standard for responsive networking systems, while showing how accessible these solutions are to implement. We also emphasize how the utilization of the relevant solutions could have aided in mitigating past threats.

In section 2, we discuss works related to the establishment of standards and techniques in networking forensics. Section 3 explains the functionality of certain freely available network security and forensics tools, as well as their potential when utilized for more responsive network architecture. Tools such as Nmap [1] and HoneyDrive3 [2] can provide actionable data for monitoring while also having the potential to act on events in an automated fashion. While tools such as Fail2ban [3] and Netdata [4] are designed for real-time network monitoring, response, and notification of unwanted activity. Section 4 provides analysis on the results from Section 3 while discussing how it applies to more responsive network forensic system approaches. Section 5 concludes the paper with final thoughts regarding implementation.

Related Work
As a subset of digital forensics, network forensics addresses the need for incident investigation after-the-fact for a fundamental structure most organizations, businesses, schools, and facilities

rely on. This makes it quite easy to see how detrimental its security may be. Information of varying worth and sensitivity must utilize this medium making it a tantalizing target. As technology changes, so too shall the threats we face. Due to the volatility of the field, we must strive for similarly volatile solutions to keep up with growing threats. Previous Researchers like [5] have discussed implementation of policies at a business level to mitigate the impact and increase deterrence of computer crimes on a network. The need for these policies was identified over a decade ago, yet we can still find examples today of networks lacking many of these networking forensics recommendations. Many of which also serve as improvements in securing the network from future attacks. Technologies like cloud computing pose a relatively new problem regarding the collection, authentication, and proof of ownership of data as discussed by [6]. Researchers have recommended possible solutions to these problems, usually in the form of newer methods of logging to be embedded in the cloud architecture like that of [7]. Yet, many of these discussions and recommendations focus heavily on what [8] would call the traditional "Reactive Network Forensics" approach characterized by reacting after an incident has occurred. This approach has proven to be less than ideal due to the cost scaling proportionally to the amount of activity logged as well as the increase in time required for an investigation that does not occur until after a crime has been committed. As we survey the following freely accessible tools we hope to convey both the importance and benefit of utilizing today's resources in what [8] refers to as the "Proactive Network Forensics" approach. This approach seeks to create a network forensics system that is prepared for an attack prior to the attack, and that is designed to identify important evidence necessary to mitigate the threat. Although we acknowledge no network can be made completely safe from threats, we believe a standard or policy advocating for a more proactive network forensic approach could greatly improve a network's security.

Experimental Work

In this section, we discuss the capabilities of each tool, the configuration, and the features tested. It should be noted that though the features of an individual tool may provide limited capability, when utilized in larger network systems along with other tools the potential for more proactive network forensics for these tools greatly increases. The following experiments were not exhaustive utilizations of all features of each tool, as the authors have chosen to focus on the evaluation of some of the more popular features of the following tools. Each subsection dedicated to tools cover specific configuration regarding the tool, and the following table shows the Operating System and computer specifications utilized for the experiments.

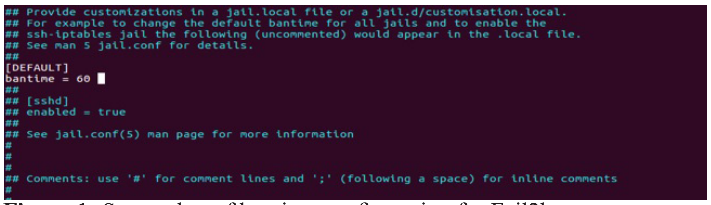
	Configuration 1	Configuration 2
OS	Windows 10 Home	Windows 10 Education
Storage	500GB HDD	500GB SSD
RAM	16 GB	8GB
Processor	Intel i7-6700HQ 2.60GHz	Intel i5-4210U 1.70GHz
System Type	64-bit	64-bit

Table 1: Computer Specifications used for testing.

Fail2ban

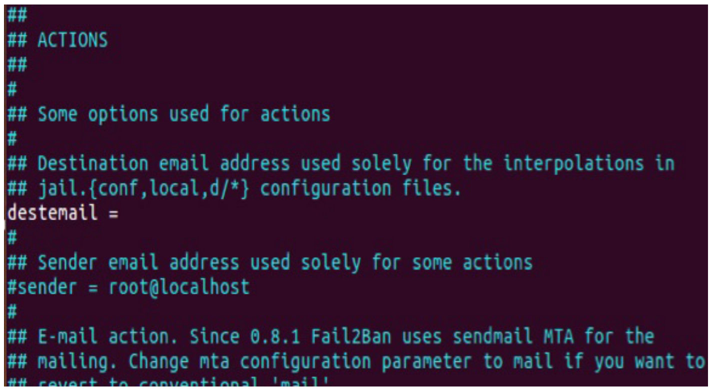
Fail2ban is a network logging tool that offers automated intrusion detection and response. It functions by actively scanning recorded activity and executes customizable responses to certain indicators of malicious actions. Once Fail2ban identifies the IPs from which these activities originate from it conducts actions as passive as sending a notification, or as active as banning the malicious machine then updating firewall rules to reject the IP address of the source machine. Configurations of Fail2ban can be modified to ban users, email addresses, or any source that is identifiable through logged activity. Actions can also be configured to react in varying severities and time periods. Default Fail2ban configuration can read standard log files from sshd and Apache, but the tool itself is designed for easy configuration to read any log file of your choosing. Included in the default configuration are well known filter-rules for things such as failed password attempts and activity through common services like ssh, apache, courier, etc. It is also worth noting all IP based features work for both IPv4 and IPv6 addresses.

For the following experimentation of Fail2ban version 0.9.6-2 configuration 2 from Table 1 was used. Current Fail2ban installation packages are only available for predominately Linux and Unix based operating systems such as Mac OS, ArchLinux, Ubuntu, Red Hat, and more. Therefore, a Virtual Machine was needed for testing the software. Oracle VM VirtualBox version 5.1.18 was used to run an Ubuntu 16.04.1 LTS VM. Installation was conducted by downloading the contents of [9] and utilizing the recommended terminal commands given to untar and install the Fail2ban tar file onto the Ubuntu VM. Since it is expected that new versions of Fail2ban are to be released in the future on the same GitHub repository, one can still access the version utilized during this research through the "Releases" page through [9]. After installation you will find multiple configuration files in your "/etc/fail2ban/" with the extension ".conf". For the experiment we began by modifying our configuration of Fail2ban. First we navigated to the "jail.conf" configuration file in the "/etc/fail2ban/" directory. It is worth noting that if you plan to maintain your version of Fail2ban and update in the future then customization of "jails", or actions you wish to occur in response to malicious activity, must be done by creating a "jail.local" under the "jail.d/" directory in order to prevent updates from erasing your changes. Since this was not a factor in our experiment we have opted to modify the default configuration file. Through this file we can configure: ban time, a "white-list" of IPs whose actions we wish to ignore, max password attempts allowed, email address to notify of malicious activity, file paths for logs, which ports to associate with which filters, and more. To set the default ban time to one minute we modified the "jail.conf" as seen below in Figure 1. The value of "bantime" was set to 60 since Fail2ban interprets this value in seconds, thus making the ban time a minute. Afterward we located the "destemail" variable, as seen in Figure 2, we can assign it to any email address we wish to be notified when our malicious activity defined by one of our filters is detected.



```
## Provide customizations in a jail.local file or a jail.d/customisation.local.
## For example to change the default bantime for all jails and to enable the
## ssh-iptables jail the following (uncommented) would appear in the .local file.
## See man 5 jail.conf for details.
##
[DEFAULT]
bantime = 60
##
## [sshd]
## enabled = true
##
## See jail.conf(5) man page for more information
#
#
#
## Comments: use '#' for comment lines and ';' (following a space) for inline comments
#
```

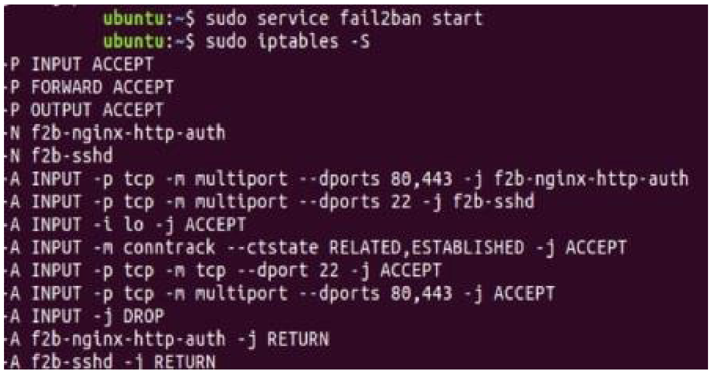
Figure 1: Screenshot of bantime configuration for Fail2ban.



```
##
## ACTIONS
##
#
## Some options used for actions
#
## Destination email address used solely for the interpolations in
## jail.{conf,local,d/*} configuration files.
destemail =
#
## Sender email address used solely for some actions
#sender = root@localhost
#
## E-mail action. Since 0.8.1 Fail2Ban uses sendmail MTA for the
## mailing. Change mta configuration parameter to mail if you want to
## revert to conventional 'mail'.
```

Figure 2: Screenshots of destemail configuration for Fail2ban.

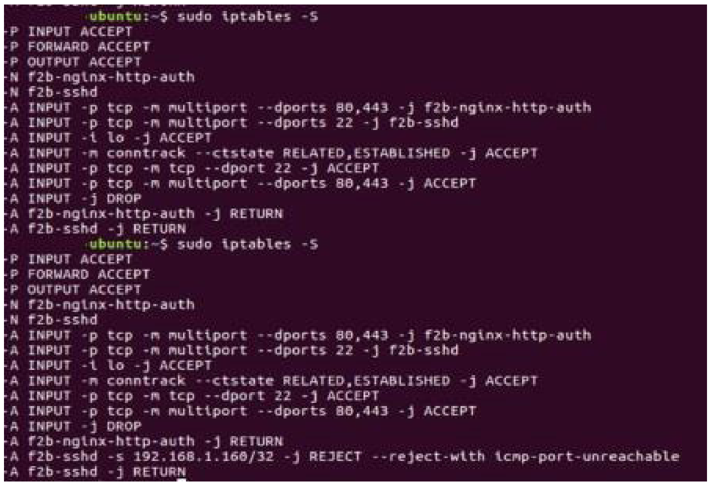
Upon configuring our destination email for notifications Fail2ban required one final configuration, which was a firewall. Since our testing environment is an Ubuntu VM we decided to utilize the standard firewall that comes with the operating system. Iptables provides standard firewall functionality by acting as a front end to the network interface. As traffic comes in a set of rules maintained and configured through the "iptables" console command is used to filter incoming traffic. In Figure 3 we show how to initiate the fail2ban service as well as the output of the "iptables –S" command which displays the rules set for our firewall during the experiment. The configuration we chose for the firewall is basic, ssh and web ports are open and incoming as well as outgoing traffic is allowed.



```
        ubuntu:~$ sudo service fail2ban start
        ubuntu:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N f2b-nginx-http-auth
-N f2b-sshd
-A INPUT -p tcp -m multiport --dports 80,443 -j f2b-nginx-http-auth
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
-A INPUT -j DROP
-A f2b-nginx-http-auth -j RETURN
-A f2b-sshd -j RETURN
```

Figure 3: Screenshot of fail2ban service start and iptable rules listing.

Next, we decided to test Fail2ban's primary goal out-of-the-box, which is to monitor login attempts and ban IP addresses belonging to machines that have exceeded the designated attempt limit. Now that we have the fail2ban service running on our Ubuntu VM, the next step is to simulate multiple failed attempts at logging into the VM remotely. Using the windows command prompt on our host machine (whose specifications can be found under configuration 2 in Table 1) we attempted to ssh into the Ubuntu VM using its IP address as the hostname and purposely failed to login three times. The fourth time we attempted to login we received an error message preventing us from attempting a fifth time, thus indicating some sort of interference. To confirm the Fail2ban service was responsible, and had updated our firewall rules to prevent further attempts we returned to the console on the Ubuntu VM and executed the "iptables –S" command to list firewall rules. In Figure 4 below the first output from Figure 3 can be seen having a noticeable difference to the most recent output, there is a new rule configured with our firewall specifically rejecting ssh attempts from an IP address which at the time was the address of the host machine. Therefore, showing that Fail2ban does as expected by actively monitoring ssh attempts to the Ubuntu VM running the Fail2ban service and banning the IP address of the machine exhibiting malicious behavior by exceeding our limit of login attempts.



```
        ubuntu:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N f2b-nginx-http-auth
-N f2b-sshd
-A INPUT -p tcp -m multiport --dports 80,443 -j f2b-nginx-http-auth
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
-A INPUT -j DROP
-A f2b-nginx-http-auth -j RETURN
-A f2b-sshd -j RETURN
        ubuntu:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N f2b-nginx-http-auth
-N f2b-sshd
-A INPUT -p tcp -m multiport --dports 80,443 -j f2b-nginx-http-auth
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
-A INPUT -j DROP
-A f2b-nginx-http-auth -j RETURN
-A f2b-sshd -s 192.168.1.160/32 -j REJECT --reject-with icmp-port-unreachable
-A f2b-sshd -j RETURN
```

Figure 4: Screenshot of new firewall rules added by Fail2ban.

Netdata

Netdata is an interactive web dashboard for "real-time performance and health monitoring systems" [4]. It is designed to require minimal maintenance and configuration, if any, after installation unless you desire the use of a specific alarm/notification medium and/or the use of adding another metric to monitor. Many APIs for messaging like Slack and Twilio are supported out-of-the-box, as well as many methods of adding metrics like python, ruby, and java. Netdata also seeks to minimize dependencies by hosting its own static web page and API on the host machine, therefore eliminating the need for any additional configuration to access live metrics other than installation. Monitoring metrics is the main feature Netdata focuses on, and arguably one could say it does it very well. Local metrics like CPU, Memory, and Network activity per interface are not the only sources of data being monitored. Interprocess communication, detailed packet analysis, metrics for DDoS detection, processes and their activity, applications, web server logistics, database queries and operations just to name a few [4]. Netdata seeks to be the universal dashboard where all metrics can be monitored. If the default installation does not already come with compatible features for measuring a desired metric, then it is likely the monitoring system can accept customized plugins for your desired metric.

Netdata is a freely available tool designed for Linux systems, therefore we will be utilizing configuration 2 from Table 1 as well as the same Ubuntu 16.04.1 LTS Virtual Machine running on VirtualBox from our testing environment used to run Fail2ban previously to conduct our tests. Consistent with the goal of the Netdata development team, installation was simple. In the “wiki” page accessible through [4], single line commands for downloading the contents of the Netdata GitHub repository or from the developer’s website can be found. These commands consist of the traditional “git clone” command to copy a repository locally, and the “curl” command to transfer data from the server locally, respectively. We used the latest version of Netdata at the time which was version 1.5.0. We attained our copy of the installation through the “curl” command accompanied by the url for the kickstart script hosted on the Netdata team’s server at the time “https://my-netdata.io/kickstart.sh”. To replicate the results of our test after the Netdata repository has been updated one can access older versions of the repository through the “Releases” page accessibly through [4]. After Netdata has been installed, open a console and navigate to the “/netdata” directory and run the “netdata-installer.sh” installer script using root privileges. A textual notification that Netdata has been installed and is running should appear in the console once the script completes. Now Netdata’s web dashboard should be live and accessible through any web browser on the system. To access the web dashboard running locally on the host machine through port 19999, simply navigate to “localhost:19999” in your web browser. A view like that of Figure 5 below, should appear.

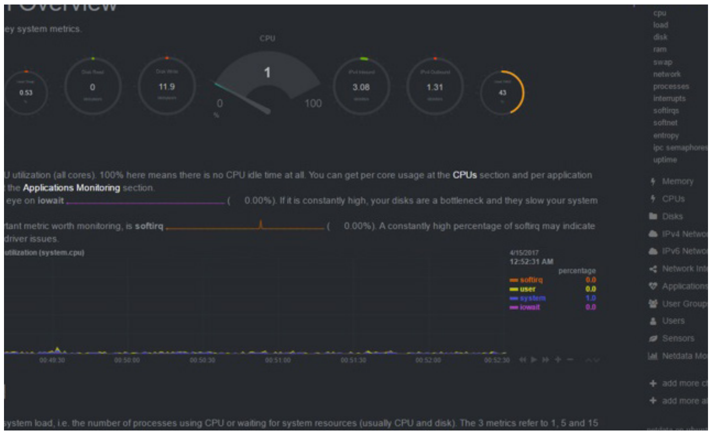

U utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the **CPUs** section and per application the **Applications Monitoring** section.
eye on **iowait** (0.00%). If it is constantly high, your disks are a bottleneck and they slow your system
tant metric worth monitoring, is **softirq** (0.00%). A constantly high percentage of softirq may indicate driver issues.
ystem load, i.e. the number of processes using CPU or waiting for system resources (usually CPU and disk). The 3 metrics refer to 1, 5 and 15

Figure 5: Screenshot of Netdata dashboard.

To test Netdata’s ability to actively monitor system data and alert us of abnormalities that could possibly signal malicious intent, we decided to run an open scan of the Ubuntu VM using Nmap. On our host machine, we used a multi-platform Nmap GUI accessible on Windows called Zenmap. In Figure 6 the parameters used to conduct the scan of our Ubuntu VM, whose IP address at the time was 192.168.247.128, are shown. Since the host machine and the Ubuntu VM were both on the same network we were capable of executing the following Nmap scan.

During this scan, an alert appeared on the Netdata dashboard indicating that an abnormal amount of TCP RESETS was being sent from the host. The alert can be seen in Figure 7 below.

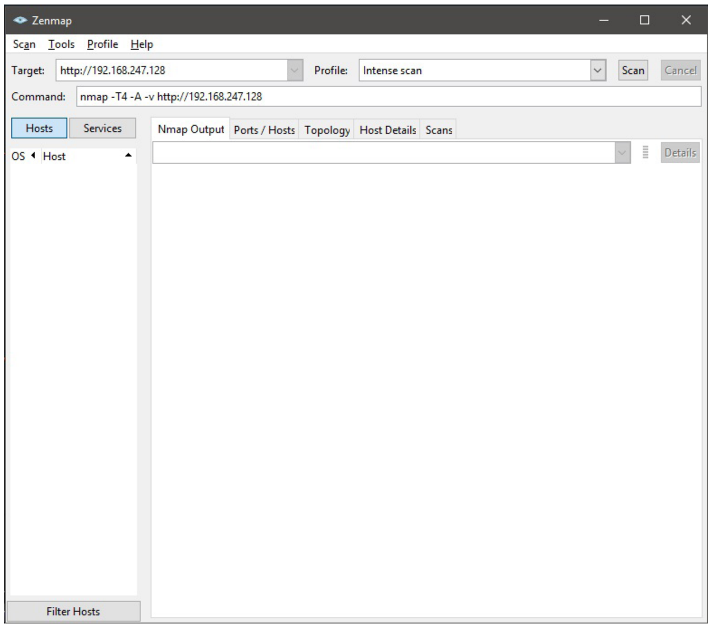
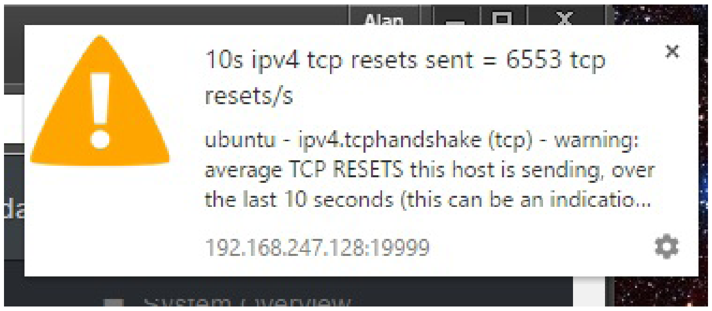

Target: http://192.168.247.128
Profile: Intense scan
Command: nmap -T4 -A -v http://192.168.247.128

Figure 6: Screenshot of Zenmap scan parameters.


10s ipv4 tcp resets sent = 6553 tcp resets/s

ubuntu - ipv4.tcphandshake (tcp) - warning: average TCP RESETS this host is sending, over the last 10 seconds (this can be an indicatio...

192.168.247.128:19999

Figure 7: Screenshot of alert given by Netdata.

We could also see a noticeable difference between metrics before and after the scan showing that Netdata does actively monitor for abnormalities in metrics like IPv4 traffic. Figure 8 shows how before the scan virtually zero IPv4 packets being received or sent by the Ubuntu VM, while Figure 9 shows an increase in IPV4 packet communication.

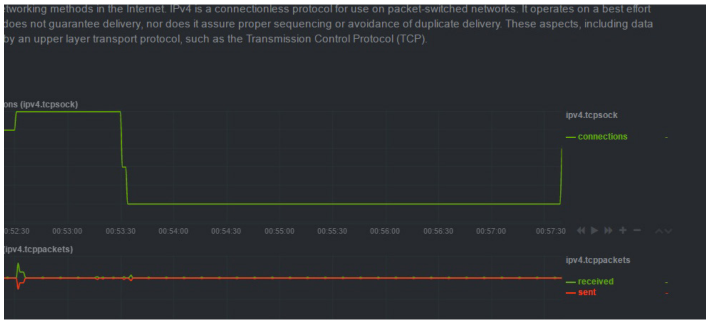

tworking methods in the Internet. IPv4 is a connectionless protocol for use on packet-switched networks. It operates on a best effort does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data by an upper layer transport protocol, such as the Transmission Control Protocol (TCP).

Figure 8: Netdata IPv4 traffic data before Nmap Scan.

It is also worth noting all abnormalities in data are logged in the “Alarm Log” accessible through the Netdata dashboard. Figure

10 shows the view of this log after the Nmap scan. As you can see not only does this view provide a simple way of checking for abnormal events, but it also indicates when the abnormalities finished and provides a "CLEAR" status indicating the activity is no longer occurring.

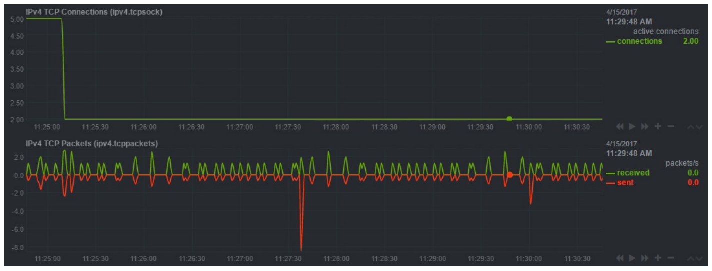
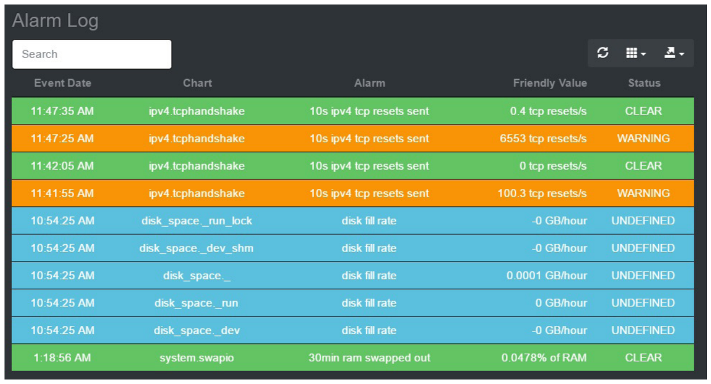


Figure 9: Netdata IPv4 traffic data after Nmap Scan.



Event Date	Chart	Alarm	Friendly Value	Status
11:47:35 AM	ipv4.tcphandshake	10s ipv4 tcp resets sent	0.4 tcp resets/s	CLEAR
11:47:25 AM	ipv4.tcphandshake	10s ipv4 tcp resets sent	6553 tcp resets/s	WARNING
11:42:05 AM	ipv4.tcphandshake	10s ipv4 tcp resets sent	0 tcp resets/s	CLEAR
11:41:55 AM	ipv4.tcphandshake	10s ipv4 tcp resets sent	100.3 tcp resets/s	WARNING
10:54:25 AM	disk_space._run_lock	disk fill rate	-0 GB/hour	UNDEFINED
10:54:25 AM	disk_space._dev_shm	disk fill rate	-0 GB/hour	UNDEFINED
10:54:25 AM	disk_space._	disk fill rate	0.0001 GB/hour	UNDEFINED
10:54:25 AM	disk_space._run	disk fill rate	0 GB/hour	UNDEFINED
10:54:25 AM	disk_space._dev	disk fill rate	-0 GB/hour	UNDEFINED
1:18:56 AM	system.swapio	30min ram swapped out	0.0478% of RAM	CLEAR

Figure 10: Netdata Alarm Log after Nmap Scan.

Nmap

Nmap is an open source "network discovery and security auditing" utility [1]. It uses a combination of raw packet data as well as creative implementations of network communication manipulation to determine information like: a target's Operating System, network mapping, security identification, port and application activity, firewall configuration, and services the target is interacting with [1]. Nmap also has a Scripting Engine capable of using Lua scripts to automate networking tasks like vulnerability detection and more sophisticated implementations of other Nmap features.

Detailed Nmap installation and documentation information can be found at [1]. For the following Nmap testing, configuration 1 from Table 1 was used. Although Nmap tools can be utilized on the host Windows machine, we felt it best to simulate a network to test Nmap giving us control over more environment variables. To simulate a network topology, we utilized GNS3 Version 1.5.3.0, a network virtualization environment for building, designing, and testing network topologies. We chose GNS3 due to it being freely accessible, team members had experience with the tool, and importing VirtualBox VMs into the topology was supported. The topology we chose was a simple network of two machines, both of which are joined by a virtual link connecting their network interfaces. Figure 11 below shows this topology.

We utilized a popular penetration testing Linux distribution known as Kali Linux version 2016.2 which comes pre-packaged with Nmap version 7.25BETA1. The Ubuntu VM is the same Ubuntu 16.04 version utilized in previous experiments. In Table 2 below you will find the network configuration given to each machine and their respective network interface using the "ifconfig" network utility that comes with Unix-like operating systems.

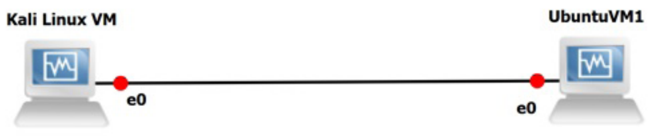
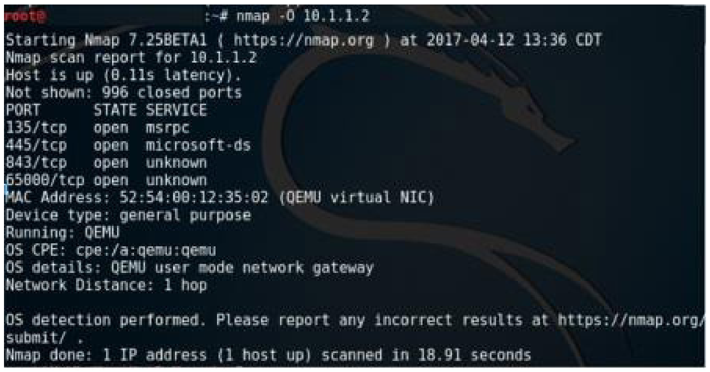


Figure 11: GNS3 network topology used for Nmap testing.

VM	IP address	Netmask	Interface
Kali Linux VM	10.1.1.1	255.255.255.0	eth0
UbuntuVM1	10.1.1.2	255.255.255.0	enp0s3

Table 2: Configuration of VMs on virtual network in GNS3.

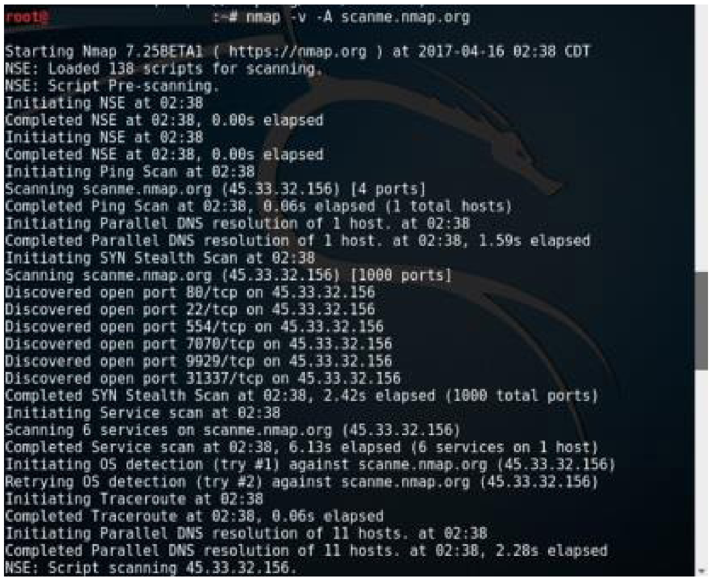
It is recommended that once the configuration is done and the GNS3 virtual environment is initiated, utilize the "ping" tool from either of the VM's consoles to ensure connectivity between the two machines. Once configuration was completed and confirmed, we started to test the features of Nmap. The first feature we tested was the operating system detection. Nmap can detect the OS of other machines on the network through TCP/IP fingerprinting. This involves the sending of TCP and UDP packets to the host, then thoroughly examining the responses by running them through TCP ISN sampling, and IP ID sampling tests. The results are then compared to known responses of multiple OSes and an answer as to what OS could have sent those responses is given. If there is uncertainty as to what OS it is exactly then multiple answers are given along with a percentage based on how much of the responses matched with that given answer's known response markers. In Figure 12 we show the result from running Nmap's OS detection in the Kali VM to determine the OS of the Ubuntu VM.



```
root@              :~# nmap -O 10.1.1.2
Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-04-12 13:36 CDT
Nmap scan report for 10.1.1.2
Host is up (0.11s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
843/tcp   open  unknown
65000/tcp open  unknown
MAC Address: 52:54:00:12:35:02 (QEMU virtual NIC)
Device type: general purpose
Running: QEMU
OS CPE: cpe:/a:qemu:qemu
OS details: QEMU user mode network gateway
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.91 seconds
```

Figure 12: Screenshot of Nmap OS detection results.

The results of the OS detection feature did not return the expected answer indicating an Ubuntu based system was found, however the results were technically accurate. Nmap detected a "QEMU" based operating system which is correct since the Ubuntu VM is running in a virtualized environment using Oracle's VirtualBox software which uses QEMU as a virtual machine emulator. This explains the QEMU virtual NIC and unexpected result. It is worth noting that the port and service information can help with determining further information about the operating system, but in this situation the information is slightly deceptive. Although Microsoft services are running on port 135 and port 445, further investigation of the diverse uses of these ports shows that non-Windows machines could also utilize these same points of service [11]. The next feature we tested was Nmap's full scan. To receive a more robust response from the scan we decided to utilize the test web address provided by the Nmap team on their website for testing purposes "scanme.nmap.org". This feature utilizes the power of multiple Nmap features at once on the given target machine. OS detection, version detection, script scanning, and tracerouting are the main features invoked when utilizing this full scan. Figure 13 through Figure 15 below show the results from this scan.

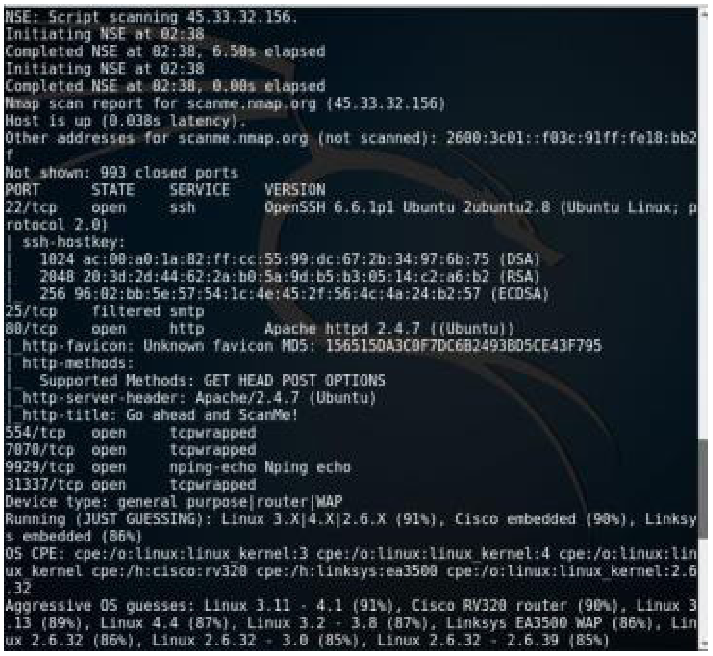


```
root@            :~# nmap -v -A scanme.nmap.org

Starting Nmap 7.25BETA1 ( https://nmap.org ) at 2017-04-16 02:38 CDT
NSE: Loaded 138 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 02:38
Completed NSE at 02:38, 0.00s elapsed
Initiating NSE at 02:38
Completed NSE at 02:38, 0.00s elapsed
Initiating Ping Scan at 02:38
Scanning scanme.nmap.org (45.33.32.156) [4 ports]
Completed Ping Scan at 02:38, 0.06s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 02:38
Completed Parallel DNS resolution of 1 host. at 02:38, 1.59s elapsed
Initiating SYN Stealth Scan at 02:38
Scanning scanme.nmap.org (45.33.32.156) [1000 ports]
Discovered open port 80/tcp on 45.33.32.156
Discovered open port 22/tcp on 45.33.32.156
Discovered open port 554/tcp on 45.33.32.156
Discovered open port 7070/tcp on 45.33.32.156
Discovered open port 9929/tcp on 45.33.32.156
Discovered open port 31337/tcp on 45.33.32.156
Completed SYN Stealth Scan at 02:38, 2.42s elapsed (1000 total ports)
Initiating Service scan at 02:38
Scanning 6 services on scanme.nmap.org (45.33.32.156)
Completed Service scan at 02:38, 6.13s elapsed (6 services on 1 host)
Initiating OS detection (try #1) against scanme.nmap.org (45.33.32.156)
Retrying OS detection (try #2) against scanme.nmap.org (45.33.32.156)
Initiating Traceroute at 02:38
Completed Traceroute at 02:38, 0.06s elapsed
Initiating Parallel DNS resolution of 11 hosts. at 02:38
Completed Parallel DNS resolution of 11 hosts. at 02:38, 2.28s elapsed
NSE: Script scanning 45.33.32.156.
```

Figure 13: Screenshot of Nmap full scan output (1 of 3).

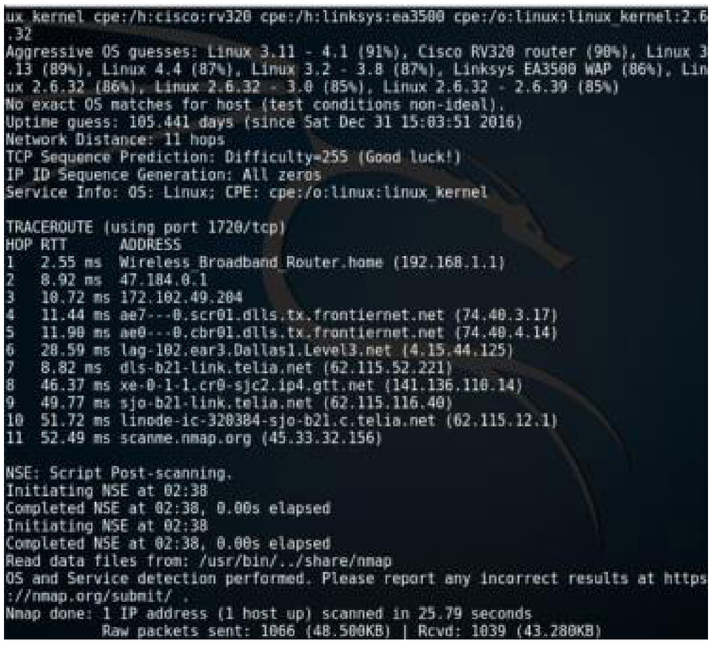
Figure 13 shows execution of default Lua scripts through Nmap's Scripting Engine (NSE) to execute multiple Nmap features in parallel. As shown in Figure 13, after the target name is resolved through a DNS server, port scanning is conducted on the target machine showing multiple open ports. This serves as one of Nmap's security auditing features since open ports when left unsupervised can pose as a vulnerability by providing remote access to the target machine itself. After port scanning has begun, other features like traceroute and OS detection are also initiated through the NSE. Figure 14 shows the results of more port scanning, including ssh hostkeys and services running on the identified ports. Towards the bottom of the screenshot we begin to see the OS detection results, in this case the responses to the Nmap packets seem to have markers like systems with Linux, Cisco, and Linksys devices. It is worth noting how the output indicates these are indeed guesses based on the responses match percentage to known OS response markers, as opposed to a definitive response.

Figure 15 below shows the remainder of the Nmap scan output. Most of which pertains to the results from the Nmap traceroute feature. The results indicating the different hops, ports utilized, and addresses reached are straightforward. As we can see the scan was successful at utilizing multiple Nmap features and applying them to a single target machine.



```
NSE: Script scanning 45.33.32.156.
Initiating NSE at 02:38
Completed NSE at 02:38, 6.50s elapsed
Initiating NSE at 02:38
Completed NSE at 02:38, 0.00s elapsed
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.038s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2
f
Not shown: 993 closed ports
PORT      STATE    SERVICE    VERSION
22/tcp    open     ssh        OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; p
rotocol 2.0)
| ssh-hostkey:
|   1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)
|   2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)
|_  256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)
25/tcp    filtered smtp
80/tcp    open     http       Apache httpd 2.4.7 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: 156515DA3C0F7DC6B2493BD5CE43F795
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Go ahead and ScanMe!
554/tcp   open     tcpwrapped
7070/tcp  open     tcpwrapped
9929/tcp  open     nping-echo Nping echo
31337/tcp open     tcpwrapped
Device type: general purpose|router|WAP
Running (JUST GUESSING): Linux 3.X|4.X|2.6.X (91%), Cisco embedded (90%), Linksy
s embedded (86%)
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 cpe:/o:linux:lin
ux_kernel cpe:/h:cisco:rv320 cpe:/h:linksys:ea3500 cpe:/o:linux:linux_kernel:2.6
.32
Aggressive OS guesses: Linux 3.11 - 4.1 (91%), Cisco RV320 router (90%), Linux 3
.13 (89%), Linux 4.4 (87%), Linux 3.2 - 3.8 (87%), Linksys EA3500 WAP (86%), Lin
ux 2.6.32 (86%), Linux 2.6.32 - 3.0 (85%), Linux 2.6.32 - 2.6.39 (85%)
```

Figure 14: Screenshot of Nmap full scan output (2 of 3).



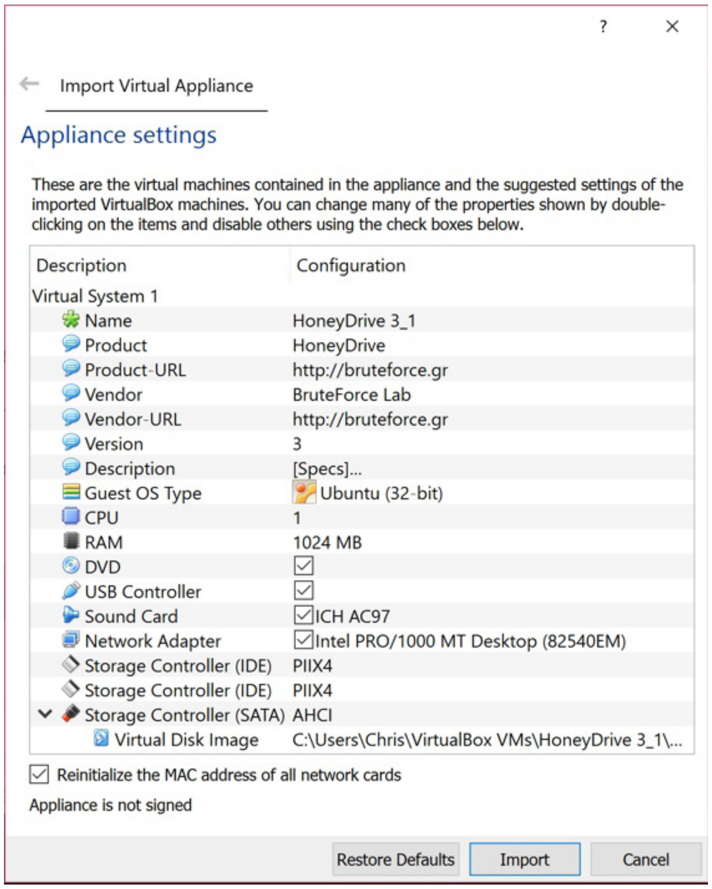
```
ux_kernel cpe:/h:cisco:rv320 cpe:/h:linksys:ea3500 cpe:/o:linux:linux_kernel:2.6
.32
Aggressive OS guesses: Linux 3.11 - 4.1 (91%), Cisco RV320 router (90%), Linux 3
.13 (89%), Linux 4.4 (87%), Linux 3.2 - 3.8 (87%), Linksys EA3500 WAP (86%), Lin
ux 2.6.32 (86%), Linux 2.6.32 - 3.0 (85%), Linux 2.6.32 - 2.6.39 (85%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 105.441 days (since Sat Dec 31 15:03:51 2016)
Network Distance: 11 hops
TCP Sequence Prediction: Difficulty=255 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 1720/tcp)
HOP RTT      ADDRESS
1   2.55 ms  Wireless_Broadband_Router.home (192.168.1.1)
2   8.92 ms  47.184.0.1
3   10.72 ms 172.102.49.204
4   11.44 ms ae7---0.scr01.dlls.tx.frontiernet.net (74.40.3.17)
5   11.98 ms ae0---0.cbr01.dlls.tx.frontiernet.net (74.40.4.14)
6   28.59 ms lag-102.ear3.Dallas1.Level3.net (4.15.44.125)
7   8.82 ms  dls-b21-link.telia.net (62.115.52.221)
8   46.37 ms xe-0-1-1.cr0-sjc2.ip4.gtt.net (141.136.110.14)
9   49.77 ms sjo-b21-link.telia.net (62.115.116.40)
10  51.72 ms linode-ic-320384-sjo-b21.c.telia.net (62.115.12.1)
11  52.49 ms scanme.nmap.org (45.33.32.156)

NSE: Script Post-scanning.
Initiating NSE at 02:38
Completed NSE at 02:38, 0.00s elapsed
Initiating NSE at 02:38
Completed NSE at 02:38, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.79 seconds
           Raw packets sent: 1066 (48.500KB) | Rcvd: 1039 (43.280KB)
```

Figure 15: Screenshot of Nmap full scan output (3 of 3).

HoneyDrive3

HoneyDrive3 is an open source honeypot linux distribution designed using Xubuntu 12.04.4 LTS [2]. It comes with 10 pre-installed/configured honeypot packages like Kippo SSH honeypot, Dionaea, Amun malware honeypot, Honeyd low-interaction honeypot, Glastopf web honeypot, Conpot SCADA/ICS honeypot, and more. Each honeypot package is designed to simulate a type of vulnerable system to attract and monitor malicious attacks conducted on it. These honeypots provide a layer of defensive between the network and attackers by leaving digital bait for the attacker to fall for. As the attacker conducts their malicious activities in this virtually sandboxed environment, those monitoring the honeypot can analyze and improve other security measures to prepare for such an attack in the future.

In this work, we test one of the more popular honeypots that HoneyDrive3 offers, Kippo SSH honeypot. Kippo comes preconfigures and ready to simulate a machine on a network. Before we can use Kippo we must first configure HoneyDrive3. Since HoneyDrive3 is a Linux distribution designed to be used as a virtual machine, we can download the preconfigured ".ova" file from [2] and open it with Oracle VM VirtualBox. For the following tests configuration 1 from Table 1 was used along with the same VirtualBox version 5.1.18 as previous tests. Upon opening the ".ova" file with VirtualBox, the appliance setting window as shown in Figure 16 should appear. We opted to reinitialize the MAC address of all network cards then import as is to avoid network configuration conflict.



These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
Virtual System 1	
Name	HoneyDrive 3_1
Product	HoneyDrive
Product-URL	http://bruteforce.gr
Vendor	BruteForce Lab
Vendor-URL	http://bruteforce.gr
Version	3
Description	[Specs]...
Guest OS Type	Ubuntu (32-bit)
CPU	1
RAM	1024 MB
DVD	✓
USB Controller	✓
Sound Card	✓ ICH AC97
Network Adapter	✓ Intel PRO/1000 MT Desktop (82540EM)
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Storage Controller (SATA)	AHCI
Virtual Disk Image	C:\Users\Chris\VirtualBox VMs\HoneyDrive 3_1\...

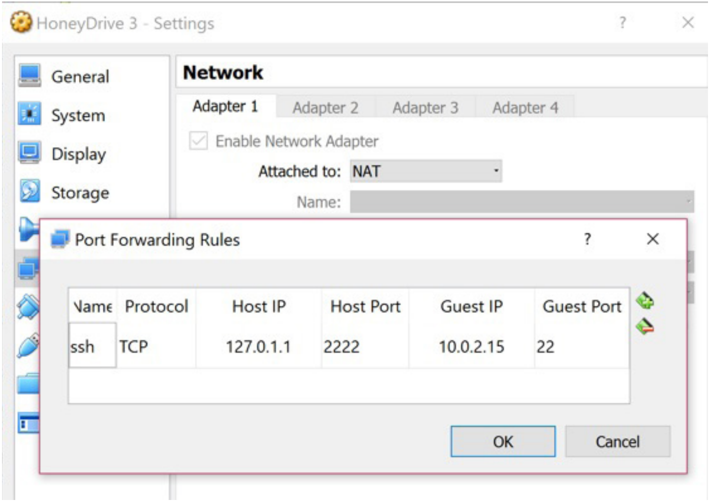
✓ Reinitialize the MAC address of all network cards

Appliance is not signed

Figure 16: Screenshot of Appliance Settings for HoneyDrive3.

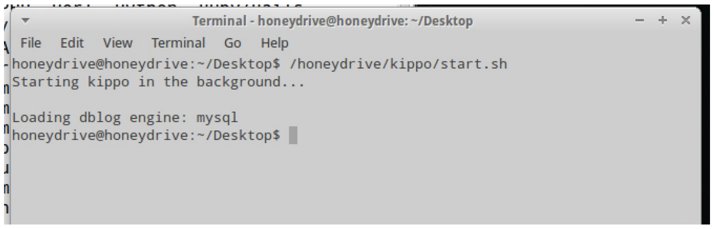
After importing the HoneyDrive3 VM, we configured the network adapter settings to attach to the host machine using NAT. To SSH from our host Windows machine to the VM we needed a way of redirecting an SSH request. Figure 17 shows how we configured a port forwarding rule in the HoneyDrive3 VM network settings.

Once this port forwarding rule has been set, HoneyDrive3 is ready to run. Information on how to start HoneyDrive's many honeypots can be found in the "README.txt" located on the desktop at start up. As indicated in the readme textfile, to start Kippo the "start.sh" script must be executed as shown in Figure 18 below.



Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
ssh	TCP	127.0.1.1	2222	10.0.2.15	22

Figure 17: Screenshot of HoneyDrive3 port forwarding rule configuration.


Terminal - honeydrive@honeydrive: ~/Desktop
File Edit View Terminal Go Help

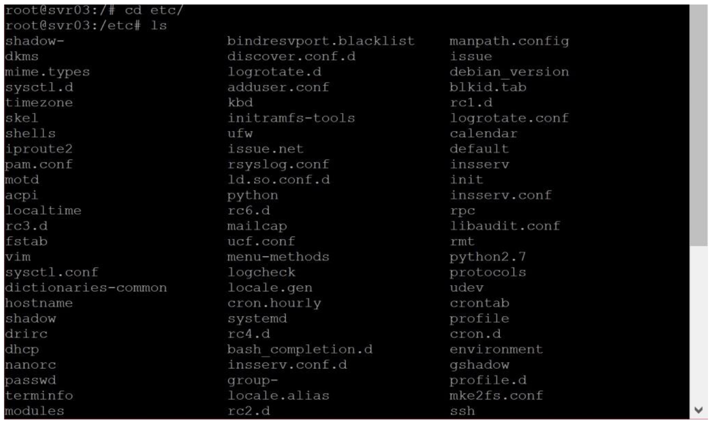
```
honeydrive@honeydrive:~/Desktop$ /honeydrive/kippo/start.sh
Starting kippo in the background...

Loading dblog engine: mysql
honeydrive@honeydrive:~/Desktop$ 
```

Figure 18: Screenshot of HoneyDrive3 Kippo start.

To ensure this process is indeed running, the "ps aux" console command accompanied with a grep for "start.sh" can be used to view a snapshot of current processes running for all users that match the chosen grep criteria. To test Kippo's ability to simulate a machine separate than that of the HoneyDrive3 VM, while also providing valuable information on the attacker's actions the next step was to SSH into the HoneyDrive3 VM. On the host PC running Windows we used PuTTY to SSH. Since the port forwarding rule from earlier indicated traffic going towards the localhost port 2222 would be redirected to the IP address of the VM port 22 where SSH attempts are expected, the SSH request through PuTTY was for 127.0.1.1:2222. Once connection is established we simulated multiple failed login attempts to root before entering the default password for the root account, which can be modified prior to running Kippo if one chooses to do so. To an attacker, the contents of the console screen once access has been attained seems normal, as shown in Figure 19 below.

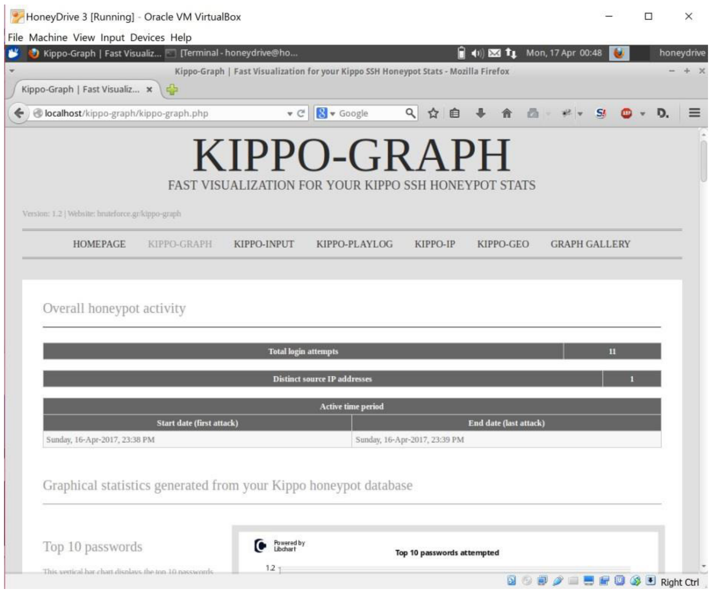
While the Kippo Honeypot service is running on the VM a local web server can be accessed to view data recorded by the honeypot. In order to view this data open a web browser on the VM and navigate to "localhost/kippo-graph/kippo-graph.php" as seen in Figure 20 below.

Through this web page visualizations of recorded data can be found. Metrics like which usernames and passwords were used, what IP addresses attempted to login, what geolocation data can be found regarding that IP address, how many successful login attempts over a designated period occurred, and even playback of recorded attacker activity can be found as shown in Figure 21.



```
root@svr03:/# cd etc/
root@svr03:/etc# ls
shadow-                       bindresvport.blacklist        manpath.config
dkms                          discover.conf.d               issue
mime.types                    logrotate.d                   debian_version
sysctl.d                      adduser.conf                  blkid.tab
timezone                      kbd                           rc1.d
skel                          initramfs-tools               logrotate.conf
shells                        ufw                           calendar
iproute2                      issue.net                     default
pam.conf                      rsyslog.conf                  insserv
motd                          ld.so.conf.d                  init
acpi                          python                        insserv.conf
localtime                     rc6.d                         rpc
rc3.d                         mailcap                       libaudit.conf
fstab                         ucf.conf                      rmt
vim                           menu-methods                  python2.7
sysctl.conf                   logcheck                      protocols
dictionaries-common           locale.gen                    udev
hostname                      cron.hourly                   crontab
shadow                        systemd                       profile
drirc                         rc4.d                         cron.d
dhcp                          bash_completion.d             environment
nanorc                        insserv.conf.d                gshadow
passwd                        group-                        profile.d
terminfo                      locale.alias                  mke2fs.conf
modules                       rc2.d                         ssh
```

Figure 19: Screenshot of Kippo SSH honeypot contents.



HoneyDrive 3 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Kippo-Graph | Fast Visualiz...
[Terminal - honeydrive@ho...
Mon, 17 Apr 00:48
honeydrive
Kippo-Graph | Fast Visualization for your Kippo SSH Honeypot Stats - Mozilla Firefox
localhost/kippo-graph/kippo-graph.php
Google
KIPPO-GRAPH
FAST VISUALIZATION FOR YOUR KIPPO SSH HONEYPOT STATS
Version: 1.2 | Website: bruteforce.gr/kippo-graph
HOMEPAGE
KIPPO-GRAPH
KIPPO-INPUT
KIPPO-PLAYLOG
KIPPO-IP
KIPPO-GEO
GRAPH GALLERY
Right Ctrl

Overall honeypot activity

Total login attempts	11
Distinct source IP addresses	1

Active time period	
Start date (first attack)	End date (last attack)
Sunday, 16-Apr-2017, 23:38 PM	Sunday, 16-Apr-2017, 23:39 PM

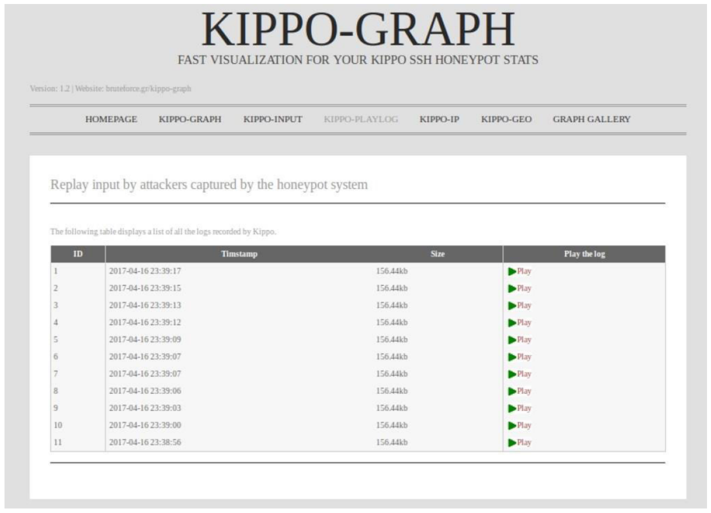
Graphical statistics generated from your Kippo honeypot database

Top 10 passwords

This vertical bar chart displays the top 10 passwords

Powered by Libchart
Top 10 passwords attempted
1.2

Figure 20: Screenshot of Kippo graph web page.



KIPPO-GRAPH
FAST VISUALIZATION FOR YOUR KIPPO SSH HONEYPOT STATS
Version: 1.2 | Website: bruteforce.gr/kippo-graph
HOMEPAGE
KIPPO-GRAPH
KIPPO-INPUT
KIPPO-PLAYLOG
KIPPO-IP
KIPPO-GEO
GRAPH GALLERY

Replay input by attackers captured by the honeypot system

The following table displays a list of all the logs recorded by Kippo.

ID	Timstamp	Size	Play the log
1	2017-04-16 23:39:17	156.44kb	▶Play
2	2017-04-16 23:39:15	156.44kb	▶Play
3	2017-04-16 23:39:13	156.44kb	▶Play
4	2017-04-16 23:39:12	156.44kb	▶Play
5	2017-04-16 23:39:09	156.44kb	▶Play
6	2017-04-16 23:39:07	156.44kb	▶Play
7	2017-04-16 23:39:07	156.44kb	▶Play
8	2017-04-16 23:39:06	156.44kb	▶Play
9	2017-04-16 23:39:03	156.44kb	▶Play
10	2017-04-16 23:39:00	156.44kb	▶Play
11	2017-04-16 23:38:56	156.44kb	▶Play

Figure 21: Screenshot of recorded playback log.

As shown, Kippo was successful at emulating a system capable of being SSH'ed into. It was also capable of recording detailed logs of our interactions while SSH'ed into the honeypot for further threat analysis.

Tools	Logging	Automated Threat Response	Active Monitoring	Capable of preventing attack	Malicious activity detection	Notifies of malicious activity	Security Auditing
Fail2ban	X	X	X		X	X	
Netdata	X		X		X	X	
Nmap	X		X				X
HoneyDrive3	X			X			

Table 3: Comparison of characteristic of tools surveyed out-of-the-box.

As shown in Table 3, many features like logging are quite common amongst network security/forensics tools. While features like automated threat response and preventative security measures to guard from attack are not as common. Even within the domain of logging some of the tools surveyed vary significantly. HoneyDrive3 provided a way of mapping geolocation data of connected hosts while providing full playback of all commands and downloads conducted while connected to the honeypot. Alternatively, there are tools like Netdata that focus on metrics like traffic, queries, and system health looking for irregularities that could signal malicious activity. Both provide information that could be detrimental to responding to threats in an accurate and timely manner, yet examples can be found today of network forensics systems that do not implement diversified logging solutions. To create more responsive network systems capable of faster and more automated threat responses both diversification of tools used as well as customization of tools must occur.

There exist multiple tools today that are open source and/or designed to allow easy customization of automated actions and threat criteria to look for. For example, Fail2ban is by default designed with the capability to block an IP in response to malicious activity originating from it, and/or notify you through email of the incident. However, as previously mentioned many of these tools have potential that exceeds their default configuration. With a preemptive network forensics system in mind, we can utilize tools like Fail2ban to not only identify customized threat criteria, but also to automatically react in a completely customizable fashion. For example, customized utilizations of Fail2ban like [10] exist showing how through the addition of an action to the "actions.d" directory along with configuration of the "jail.local" file one can have Fail2ban run a custom script utilizing an api in response to any specified log activity.

Conclusion and Future Work

This work addresses the most popular four network security tools, namely Fail2ban, Netdata, Nmap, and HoneyDrive3, and uses them to test run on an experimental setup to demonstrate how each tool can be used for network forensics purposes. The results we obtained give insight that each tool is a safe resource for utilizing as a network forensics tool as they achieve the goal of detecting an anomaly/intrusion successfully in a timely manner. Moreover, we compare these tools with respect to 7 fundamental forensics criteria as logging, automated threat response, active monitoring, capability to prevent attack, malicious activity detection, notification of malicious activity, and security auditing. Results show that Fail2ban outperforms the rest of the tools by supporting 5 out of 7 of the forensics criteria. It is followed by Netdata, which meets 4 out of 7 criteria. The remaining two tools Nmap and HoneyDrive3 performs the least as compared to other two tools by only meeting 3 out of 7 forensics criteria, though different ones.

Our work can be used as a guideline to determine which tool to use for network forensics purposes with respect to different criteria considered in our experiments.

We plan on extending our experiments to include more tools so as to provide a wider span of analysis in the field and help potential tool users determine which tool to employ whenever needed.

Acknowledgement

The authors would like to thank Mr. Alan Padilla for his contribution to the experimental work in this paper.

References

1. https://nmap.org
2. https://sourceforge.net/projects/honeydrive/?source=typ_redirect
3. https://www.fail2ban.org/wiki/index.php/Fail2Ban
4. https://github.com/firehol/netdata/wiki
5. Alec Yasinac, Yanet Manzano. Policies to Enhance Computer and Network Forensics in Proc. of the 2001 IEEE Workshop on Information Assurance and Security. 2001; IEEE.
6. Josiah Dykstra and Alan T. Sherman. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques in Digital Investigation. 2012; © Dykstra & Sherman.
7. Alecsandru Pătrașcu, Victor-Valeriu Patriciu. Logging System for Cloud Computing Forensic Environments in Control Engineering and Applied Informatics. Bucharest, Romania. 2014; 16: 80-88.
8. Gulshan Shrivastava. Network forensics: Today and tomorrow, in 2016 Int Conf on Computing for Sustainable Global Development. 2016; IEEE.
9. https://github.com/fail2ban/fail2ban
10. https://github.com/toonketels/fail2ban-sms
11. http://www.speedguide.net/ports.php