Raspberry Pi as an Intrusion Detection System, a Honeypot and a Packet Analyzer

Shyava Tripathi
Department of Computer Science Engineering, Amity School of Engineering & Technology, Amity University
Noida, Uttar Pradesh, India
shyava.tripathi8@gmail.com

Rishi Kumar
Assistant Professor, Department of Computer Science Engineering, Amity School of Engineering & Technology, Amity University, Noida, Uttar Pradesh, India
rkumar25@amity.edu

Abstract—***With cybercrimes and attacks on the rise, security is a basic requirement, as well as a necessity, in today's world. As any type of intercommunication and storage of data on the internet is becoming precarious, it becomes extremely important to keep our home network well updated in terms of security, minimize loop holes and ensure data integrity and incorruptibility. It is now extremely important that we protect and secure our home networks and devices from cyber-attacks. Intrusion Detection Systems and Honeypots if implemented correctly can prove to be efficient solutions. In this paper, we evaluate the performance of a Raspberry Pi module running an IDS or Intrusion Detection System, a packet analyzer and a decoy server, called honeypot, for complete network monitoring and security.***

Keywords-Raspberry Pi, Intrusion Detection System, Honeypot, Packet Analyser, Security

I. Introduction

With cybercrimes and attacks on the rise, security is a basic requirement, as well as a necessity, in today's world. As any type of intercommunication and storage of data on the internet is becoming precarious, it becomes extremely important to keep our home network well updated in terms of security, minimize loop holes and ensure data integrity and incorruptibility. Furthermore, with the evolution of IOT or the Internet of Things, our simple home networks have gradually matured into complex smart networks making them more susceptible to attacks [1][13]. This project is devoted to implement an all-in-one defensive Raspberry Pi that will secure a computer network by acting as an IDS and filter malicious packets as well as work as a packet analyzer and a decoy honeypot server to record and analyze attacks. The objective for this research and project is to comprehend the significance of network monitoring devices and to propose a cost effective network security tool. The Raspberry Pi's portable form factor, inexpensive cost, and decent processing power are the reasons that it has been chosen as the target platform. An intrusion detection system surveils network traffic and generates alerts if any dubious data-packets are intercepted while honeypots imitate machines that hackers want to attack, with the aim of deflecting attacks and researching malicious intent by capturing details about the attack and the attacker. A packet analyzer performs real time analysis of log traffic passing through and from a network [2]. A small-scale network security solution, such as this, with easy implementation and inexpensive maintenance cost, can help strengthen security in home networks and small businesses. The project utilizes various open source software packages, installed on a third generation Raspberry Pi for network monitoring and analysis. For Instance, we are using Snort to configure the IDS where Snort is the sensor module that provides raw traffic analysis and packet monitoring [3]. To establish a packet analyzer on the pi, Tshark is configured, which is a widely used network protocol analyzer capable of sniffing network packets and displaying them in human-readable format. Lastly, we configure Cowrie as the honeypot. In this ever-evolving world of global data communication, rapid data generation and inexpensive internet connection, it is of utmost importance that we prioritize home network security in today's world or we may become exposed to attackers who strive to exploit our digital lifestyle for malicious purposes. A network monitoring tool comprising of an IDS, Honeypot and packet analyzer can be an extremely advantageous add-on to one's home network.

II. THEORETICAL CONCEPTS

A Raspberry Pi, in the simplest terms, is a **credit-card sized computer** devised by Eben Upton. Upton's motivation, the 1981 BBC Micro led to the development of this project which was designed to promote basic computer science education in schools and universities. It was developed to provide the masses with an inexpensive device that could ameliorate programming and hardware knowledge among the youth. Undeniably, the Raspberry Pi's processing power is much less than a modern computer but is nevertheless a portable, low-cost, computer based on Linux, capable of rendering all the programs that operate at a low-power consumption level [4]. Figure 1 shows the architecture of the Raspberry Pi 3B model. An economical, portable mini-computer that can be plugged into an output screen, along with a standard mouse and keyboard to function entirely as a normal laptop or desktop computer. It is an efficient little device which is proficient at doing everything you'd intend to do on a regular computer, from skimming through the web, playing high definition video games, robotics, to developing cost and power efficient home automation devices.

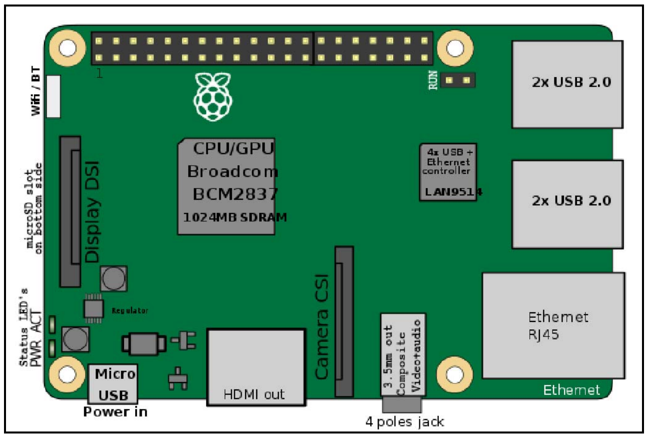


Figure 1: Raspberry Pi 3B board

This device also has the ability to connect with the outside world and is extensively being used in a variety of projects in almost every field. The various concepts related to the proposed system are as follows:

1. Intrusion Detection System
2. Honeypot
3. Packet Analyzer

2.1 Intrusion Detection System(IDS)

An Intrusion Detection System can be defined as a security application for computers and networks which is capable of accumulating and analyzing data by intercepting inbound-outbound network traffic and detecting malicious activities using specific rules, followed by alerting the user. Intrusion detection systems are an integral part of network security as they effectuate the defense in depth strategy on networks by identifying and mitigating attacks followed by logging data for future analysis.
An IDS can be host-based or network-based [5]:

- **Host-Based IDS**: An IDS which is deployed on a host machine and scans the host machine traffic to capture malicious packets and log attacks.
- **Network-Based IDS**: An IDS which is deployed on a physical network and intercepts all the traffic being sent to the network to detect abnormal packets and log attacks.

An IDS uses two ways to detect malicious activities, Anomaly-Based and Rule-Based [6]:

- **Anomaly-Based**: Anomaly based or behavior-based detection monitors network, hosts and users over time to generate data and then generate alerts when unusual or abnormal traffic is intercepted. It deploys rules based on previous attacks.
- **Rule-Based**: Rule-based or signature-based detection uses predetermined rules and compares them with current traffic to identify an attack. It uses rules written by security experts.

2.2 Honeypot

A honeypot is deployed as a decoy system setup to lure attackers and collect information about the attack as well as the attacker. In simple terms, a honeypot emulates systems that may have the potential to be attacked by hackers. The main objective of a honeypot is to deflect attacks and accumulate data about the attack for further analysis. To be able to study and analyze the attack methods helps in future optimizations to a great extent.
Honeypots function as traps set up to lure attackers by exhibiting vulnerabilities and pretending to be a normal system. They are able to log attack details with the help of logging and monitoring softwares. An unused IP address in given to the honeypot, and it is constantly under surveillance by the administrator. Honeypot will wait for an interaction to be initiated with the system, and any kind of interaction with the honeypot is regarded as dubious. They can very well waste the hacker's time, if not completely deflect the attack and can be highly effective if deployed properly.
A honeypot when deployed in a network will accumulate as much data as possible about the attack, which when analyzed will help in studying and removing system vulnerabilities, thus removing any security loop holes and keeping the system safe. They can further be combined with alert generating and output logging softwares. A honeypot can be a research honeypot or a productive honeypot, based on its deployment.
Based on the area of deployment, honeypots can be categorized as research honeypots and production honeypots.

- **Research Honeypot**: Extracting and logging information about the attack and attackers is the only function performed by a research honeypot.
- **Production Honeypot**: Along with extracting and logging information about an attack, production honeypots also provide real time security by slowing down an attack.

Honeypots can be further classified as low-interaction, medium-interaction and high-interaction honeypots.

- **Low-Interaction Honeypot**: These honeypots are called low interaction because they provide only a subset of emulated services to detect malware. For example, Dionaea is a low-interaction honeypot, which emulates Windows-2000 system vulnerabilities.
- **Medium-Interaction Honeypot**: The honeypots which are capable of fooling the attacker by pretending to host a false operating system and thus trapping the attacker are called medium-interaction honeypots. For instance, Cowrie is a medium-interaction honeypot that emulates SSH shell.
- **High-Interaction Honeypot**: These are advanced honeypots and can be extremely complex to setup as they host their own operating system. Such honeypots allow the attacker to interact with the decoy machine operating system like they would have with a regular operating system, thus capturing maximum attack details.

For example, honeynet, which is a combination of multiple decoy honeypots of different interaction levels.

2.3 Packet Analyzer

A network packet analyzer works by capturing real time network traffic and further analysing and decoding it to provide meaningful data that can be utilized to diagnose underlying network problems. They monitor the inbound and outbound network traffic and show them in a human readable format. A packet sniffer could be a dedicated hardware device deployed within a network or an embedded software application running on a standalone computer.

Analyzers can supplement other network monitoring tools such as firewalls and anti-malwares and plays a crucial role in network management and security. However, analyzers can be deployed both legally and illegally. An analyzer used to provide network security, deployed with the permission from the owner to provide diagnostics and manage a network will be termed as a legal sniffer, however, analyzers used by hackers to gain unauthorized access with the intent of spying or stealing sensitive data without the permission from the owner is termed as an illegal sniffer. Most packet analyzers aren't detected as they passively collect the network data, however, the active analyzers are likely to get detected.

Packet analyzers intercept and log network traffic that is accessible to them on the network interface they are deployed on. The data accessible to the analyzer depends on the type of network it is configured to. In a wired network, the data captured by the analyzer depends on the structure of that network, depending on the configuration of network switches. In a wireless network, analyzers can only capture data from one channel at a time unless multichannel capture has been enabled on multiple network interfaces. Once the data has been intercepted, it must be presented in a human readable format for analysis. Data is interpreted to ascertain faults and vulnerabilities, such as determining failed network requests et-cetera.

III. Proposed System

The basic setup of the proposed system consists of a Raspberry Pi 3 module connected to the Wi-Fi router with a LAN cable. It is powered on using a micro-usb cable. The raspberry pi memory card has been boot-loaded with a Raspbian-Stretch-2018-Desktop-Image. A monitor has been connected to the raspberry pi using the provided HDMI port for output and a keyboard and mouse have been attached to USB2.0 ports for input.

The intrusion detection system, the packet analyzer and the honeypot server will be deployed between the router access-point and the public network, as shown in figure 2. We have configured a network-based intrusion detection system, a packet sniffer-analyzer as well as a decoy honeypot.

The Raspberry Pi has been given dhcp configuration for the eth0 interface, and the interface eth0 has been enabled on boot.

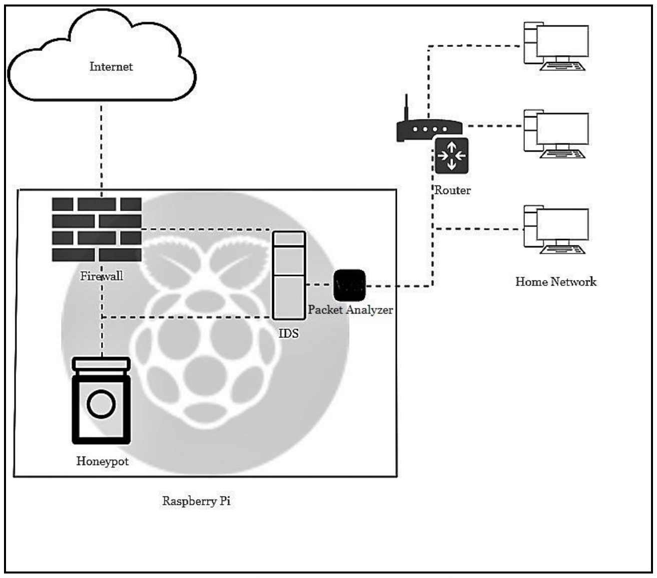


Figure 2: Architecture of proposed system

3.1 Raspberry Pi as an Intrusion Detection System

We are configuring a rule-based Network Intrusion Detection System on the pi.

We are using the following Open Source softwares to configure our IDS:

- **Snort**

Snort is one of the most expansively utilized and widely deployed open source network intrusion detection softwares devised to provide real-time analysis and sniffing of data packets on communication networks [7]. Now owned by Cisco Systems, it was initially developed by Martin Roesch in 1998. Snort is capable of functioning as a packet-analyzer, a pre-processor as well as a detection engine-logger but best of all, snort is free and compatible with most of the platforms. The data packets are captured in real-time and saved in the 'tcpdump' format for further analysis. Snort is reckoned to have an enormous data set of signatures for mischievous exploits. It looks for a fixed set of data in the packet stream depending on the configured rule set and signatures and reports them. The packets sniffed from the network are decoded by the Packet-Decoder which are further forwarded to the pre-processor whose function is to alter them to match the requirements of the detection engine. The detection engine examines the packets for the presence of any abnormal activity. Malicious packets are further logged by the logging system.

Rule Used: Snort rule is constructed to generate an alert whenever an external IP address (IPs from outside home network) attempts to initiate an ICMP Ping [8] [9].

```
alert icmp any any -> $HOME_NET any
(msg:"ICMP test detected"; GID:1; sid:10000001;
rev:001; classtype: icmp-event;)
```

▪ **Barnyard2**

Barnyard2 is an output module for snort and processes the alerts generated by snort into a database format. It's an Open Source interpreter for snort and helps in data processing. Snort saves the sniffed data to a log file in a binary format, called unified2. The function of barnyard2 is to gather this data and parse it to a readable format followed by sending it to various databases for output logging. Input processors and output plugins are the 2 major types of directives. Input processors read data from the unified2 file format followed by logging output by output plugins.

▪ **Pulledpork**

Pulledpork is a rule management tool based on PERL for intrusion detection systems like Snort and Suricata. It can be used to detect you Snort version and automatically download an appropriate rule set for you. The name 'PulledPork' was chosen simply because this application pulls the latest rules.
The interaction between Snort, Pulledpork and Barnyard3 is represented in figure 3.

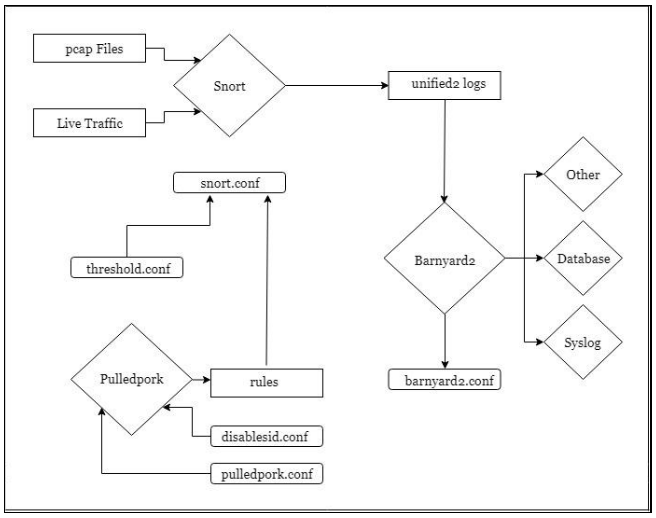


Figure 3: Snort, Pulledpork and Barnyard2 Interaction

3.2 Raspberry Pi as Honeypot

For our Raspberry Pi honeypot, we are configuring a medium-interaction SSH honeypot, Cowrie, used to log complete shell interaction during attack as well as SSH and brute force attacks.

▪ **Cowrie**

It is a python based, medium interaction honeypot which can very efficiently record the entire shell interaction during an attack, along with being able to log brute force attempts, termed as brute force cracking. This is a trial and error-based procedure used by attackers to crack passwords and paraphrases by repetitive exhausting. Cowrie enables an attacker to initiate a connection and login to the system, fooling them to believe that they are entering the legit system SSH session. Once the brute-force attack to crack the password is successful, the attacker is redirected to the decoy operating system with which they can interact. Once the attacker enters the decoy system, the system mimics the actual system and can monitor and log all the malicious actions performed by the attacker, as shown in figure 4. Cowrie offers fake file systems with the potential to add and delete files for effective mimicry. It also stores all the files that were downloaded during the interaction.

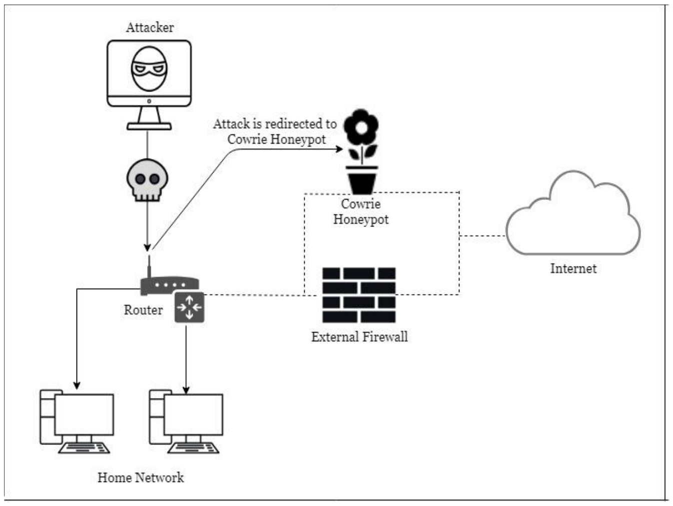


Figure 4: Cowrie honeypot network setup

3.3 Raspberry Pi as Packet Analyzer

To setup the Raspberry Pi as a network sniffer and analyzer, we will configure Tshark, a command line version of Wireshark for packet capture and analysis.

▪ **Tshark**

Tshark is a terminal based network monitoring and analysis tool used for real time network traffic analysis. It is a network monitoring tool deployed to intercept and analyze the data traffic over communication channels. It allows you to collect real time traffic data or analyze packets from previously stored traffic logs and presents it in a simplified easy to interpret format [10]. The capture file format used by Tshark is pcap and tcpdump. Tshark will work like tcpdump if no options have been set.

Packet sniffers can serve as an imperative tool for network security and monitoring. They are used to monitor network usage, for example, identifying if a network is congested and discerning bottlenecks. Moreover, they are used to identify network problems and detecting security loopholes, for instance, in case of a network error, they are used to identify the source of this error [11]. Tshark captures packets which in turn reveal packet information, layer information, free memory information as well as provides graphical representation of all the above information, as depicted in figure 5.

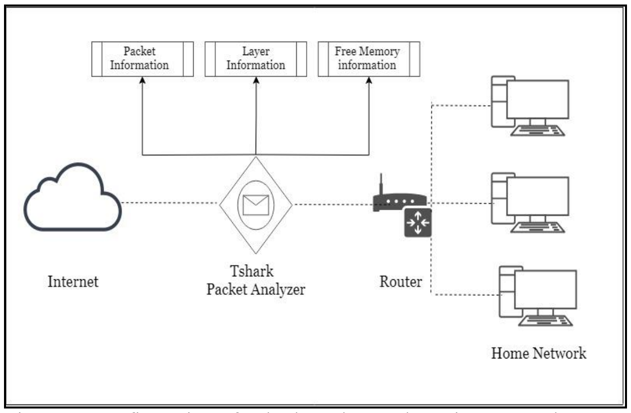


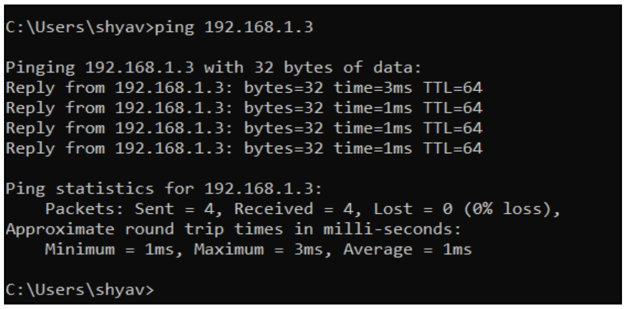
Figure 5: Configuration of Tshark packet analyzer in a network

IV. Results

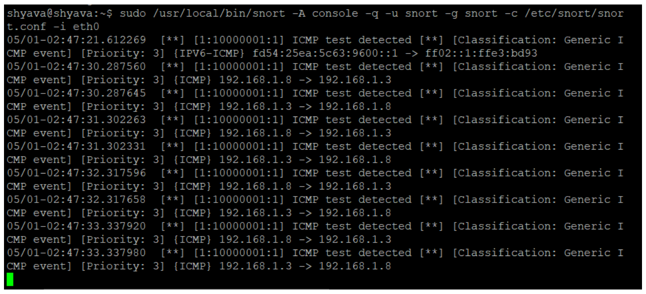
To successfully test our proposed IDS, Packet Analyzer and honeypot system, it was ensured that Snort, Barnyard2, PulledPork, MySQL, Ruby, Apache2, TShark, and Cowrie were successfully running.

4.1 Intrusion Detection System Output Analysis

The Raspberry Pi was pinged with an attacking machine to check if snort was detecting and logging alerts. Figure 6 shows the ping command executed by the attacker machine.


```
C:\Users\shyav>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:
Reply from 192.168.1.3: bytes=32 time=3ms TTL=64
Reply from 192.168.1.3: bytes=32 time=1ms TTL=64
Reply from 192.168.1.3: bytes=32 time=1ms TTL=64
Reply from 192.168.1.3: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 1ms

C:\Users\shyav>
```
Figure 6: Screenshot of ICMP ping performed by Attacker Machine

The Raspberry Pi terminal logged the ICMP details, as shown in figure 7. Snort Intrusion Detection System was successfully able to log ICMP ping requests generated by the attacker machine.

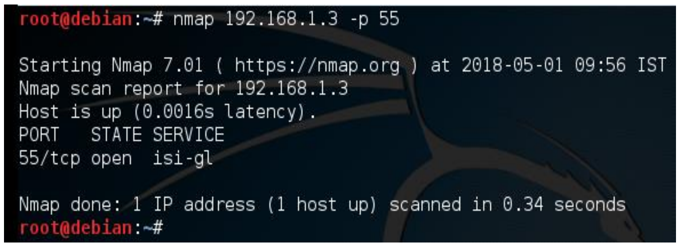

```
shyava@shyava:~$ sudo /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort/snor
t.conf -i eth0
05/01-02:47:21.612269  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {IPV6-ICMP} fd54:25ea:5c63:9600::1 -> ff02::1:ffe3:bd93
05/01-02:47:30.287560  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.3
05/01-02:47:30.287645  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.3 -> 192.168.1.8
05/01-02:47:31.302263  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.3
05/01-02:47:31.302331  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.3 -> 192.168.1.8
05/01-02:47:32.317596  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.3
05/01-02:47:32.317658  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.3 -> 192.168.1.8
05/01-02:47:33.337920  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.8 -> 192.168.1.3
05/01-02:47:33.337980  [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic I
CMP event] [Priority: 3] {ICMP} 192.168.1.3 -> 192.168.1.8
```
Figure 7: Screenshot of ICMP Ping Detected by the Raspberry Pi

4.2 Honeypot Output Analysis

To test Cowrie, we will perform an nmap scan, followed by a brute force attack performed from Kali-Linux computers.

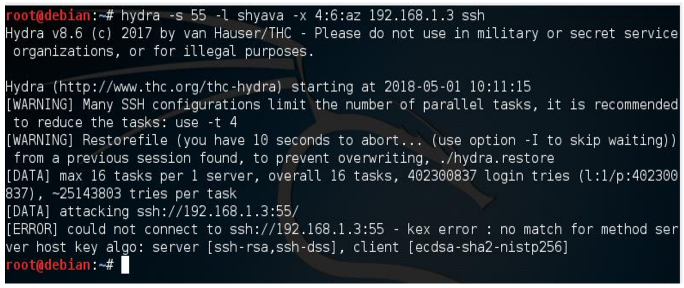
Nmap Scan:

It is performed to look for open-port vulnerabilities in the target system, as shown in figure 8.

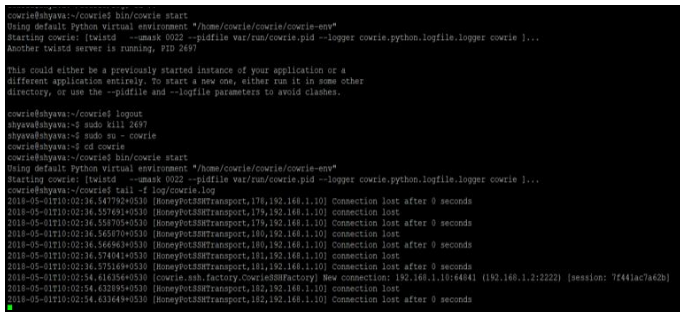

```
root@debian:~# nmap 192.168.1.3 -p 55

Starting Nmap 7.01 ( https://nmap.org ) at 2018-05-01 09:56 IST
Nmap scan report for 192.168.1.3
Host is up (0.0016s latency).
PORT   STATE SERVICE
55/tcp open  isi-gl

Nmap done: 1 IP address (1 host up) scanned in 0.34 seconds
root@debian:~#
```
Figure 8: Nmap Output

Now, we will try to access the open ports using SSH terminal followed by brute-forcing the password.

A Brute Force Attack uses trial and error method to gather sensitive user information such as username, passwords, PINs et-cetera which is then used to gain unauthorized access, as depicted in figure 9.


```
root@debian:~# hydra -s 55 -l shyava -x 4:6:az 192.168.1.3 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service
 organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-05-01 10:11:15
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended
 to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting))
 from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 402300837 login tries (l:1/p:402300
837), ~25143803 tries per task
[DATA] attacking ssh://192.168.1.3:55/
[ERROR] could not connect to ssh://192.168.1.3:55 - kex error : no match for method ser
ver host key algo: server [ssh-rsa,ssh-dss], client [ecdsa-sha2-nistp256]
root@debian:~# 
```
Figure 9: Brute-Force Attack Attempt

After performing the attack, cowrie logs were checked and as shown in figure 10, Cowrie was successful in detecting and logging an SSH Brute force attack.

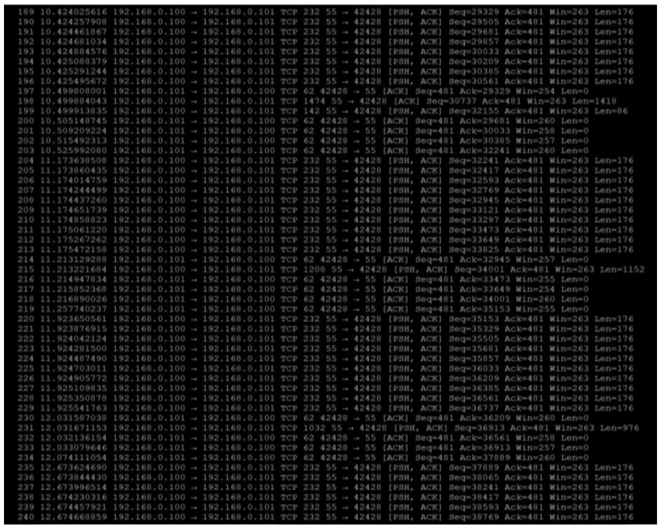

```
cowrie@shyava:~/cowrie$ bin/cowrie start
Using default Python virtual environment "/home/cowrie/cowrie/cowrie-env"
Starting cowrie: [twistd   --umask 0022 --pidfile var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie ]...
Another twistd server is running, PID 2697

This could either be a previously started instance of your application or a
different application entirely. To start a new one, either run it in some other
directory, or use the --pidfile and --logfile parameters to avoid clashes.

cowrie@shyava:~/cowrie$ logout
shyava@shyava:~$ sudo kill 2697
shyava@shyava:~$ sudo su - cowrie
cowrie@shyava:~$ cd cowrie
cowrie@shyava:~/cowrie$ bin/cowrie start
Using default Python virtual environment "/home/cowrie/cowrie/cowrie-env"
Starting cowrie: [twistd   --umask 0022 --pidfile var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie ]...
cowrie@shyava:~/cowrie$ tail -f log/cowrie.log
2018-05-01T10:02:36.547792+0530 [HoneyPotSSHTransport,178,192.168.1.10] Connection lost after 0 seconds
2018-05-01T10:02:36.557691+0530 [HoneyPotSSHTransport,179,192.168.1.10] connection lost
2018-05-01T10:02:36.558705+0530 [HoneyPotSSHTransport,179,192.168.1.10] Connection lost after 0 seconds
2018-05-01T10:02:36.565870+0530 [HoneyPotSSHTransport,180,192.168.1.10] connection lost
2018-05-01T10:02:36.566963+0530 [HoneyPotSSHTransport,180,192.168.1.10] Connection lost after 0 seconds
2018-05-01T10:02:36.574041+0530 [HoneyPotSSHTransport,181,192.168.1.10] connection lost
2018-05-01T10:02:36.575169+0530 [HoneyPotSSHTransport,181,192.168.1.10] Connection lost after 0 seconds
2018-05-01T10:02:54.616356+0530 [cowrie.ssh.factory.CowrieSSHFactory] New connection: 192.168.1.10:64841 (192.168.1.2:2222) [session: 7f441ac7a62b]
2018-05-01T10:02:54.632895+0530 [HoneyPotSSHTransport,182,192.168.1.10] connection lost
2018-05-01T10:02:54.633649+0530 [HoneyPotSSHTransport,182,192.168.1.10] Connection lost after 0 seconds
```
Figure 10: SSH Brute Force Attack detected on the Raspberry Pi

4.3 Packet Analyzer Output Analysis

The packet analyzer Tshark is started with the following commands to capture network traffic of all kinds.

```
root Debian:~# sudo tshark -i any
```

Figure 11 shows all the network traffic data packets successfully captured by Tshark.



```
189 10.424025616 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=29329 Ack=481 Win=263 Len=176
190 10.424257908 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=29505 Ack=481 Win=263 Len=176
191 10.424461867 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=29681 Ack=481 Win=263 Len=176
192 10.424681034 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=29857 Ack=481 Win=263 Len=176
193 10.424884576 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=30033 Ack=481 Win=263 Len=176
194 10.425088379 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=30209 Ack=481 Win=263 Len=176
195 10.425291244 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=30385 Ack=481 Win=263 Len=176
196 10.425495672 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=30561 Ack=481 Win=263 Len=176
197 10.499808001 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=29329 Win=254 Len=0
198 10.499884043 192.168.0.100 → 192.168.0.101 TCP 1474 55 → 42428 [ACK] Seq=30737 Ack=481 Win=263 Len=1418
199 10.499913835 192.168.0.100 → 192.168.0.101 TCP 142 55 → 42428 [PSH, ACK] Seq=32155 Ack=481 Win=263 Len=86
200 10.505148745 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=29681 Win=260 Len=0
201 10.509209224 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=30033 Win=258 Len=0
202 10.515492313 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=30385 Win=257 Len=0
203 10.525992080 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=32241 Win=260 Len=0
204 11.173638508 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=32241 Ack=481 Win=263 Len=176
205 11.173860435 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=32417 Ack=481 Win=263 Len=176
206 11.174014759 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=32593 Ack=481 Win=263 Len=176
207 11.174244499 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=32769 Ack=481 Win=263 Len=176
208 11.174437260 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=32945 Ack=481 Win=263 Len=176
209 11.174651739 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=33121 Ack=481 Win=263 Len=176
210 11.174858823 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=33297 Ack=481 Win=263 Len=176
211 11.175061220 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=33473 Ack=481 Win=263 Len=176
212 11.175267262 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=33649 Ack=481 Win=263 Len=176
213 11.175472158 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=33825 Ack=481 Win=263 Len=176
214 11.213129288 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=32945 Win=257 Len=0
215 11.213221684 192.168.0.100 → 192.168.0.101 TCP 1208 55 → 42428 [PSH, ACK] Seq=34001 Ack=481 Win=263 Len=1152
216 11.214947834 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=33473 Win=255 Len=0
217 11.215852368 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=33649 Win=254 Len=0
218 11.216890026 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=34001 Win=260 Len=0
219 11.257740237 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=35153 Win=255 Len=0
220 11.923650561 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=35153 Ack=481 Win=263 Len=176
221 11.923876915 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=35329 Ack=481 Win=263 Len=176
222 11.924042124 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=35505 Ack=481 Win=263 Len=176
223 11.924281500 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=35681 Ack=481 Win=263 Len=176
224 11.924487490 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=35857 Ack=481 Win=263 Len=176
225 11.924703011 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=36033 Ack=481 Win=263 Len=176
226 11.924905772 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=36209 Ack=481 Win=263 Len=176
227 11.925109835 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=36385 Ack=481 Win=263 Len=176
228 11.925350878 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=36561 Ack=481 Win=263 Len=176
229 11.925541763 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=36737 Ack=481 Win=263 Len=176
230 12.031587038 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=36209 Win=260 Len=0
231 12.031671153 192.168.0.100 → 192.168.0.101 TCP 1032 55 → 42428 [PSH, ACK] Seq=36913 Ack=481 Win=263 Len=976
232 12.032136154 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=36561 Win=258 Len=0
233 12.033079646 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=36913 Win=257 Len=0
234 12.074111054 192.168.0.101 → 192.168.0.100 TCP 62 42428 → 55 [ACK] Seq=481 Ack=37889 Win=260 Len=0
235 12.673624690 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=37889 Ack=481 Win=263 Len=176
236 12.673844430 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=38065 Ack=481 Win=263 Len=176
237 12.673996514 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=38241 Ack=481 Win=263 Len=176
238 12.674230316 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=38417 Ack=481 Win=263 Len=176
239 12.674457921 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=38593 Ack=481 Win=263 Len=176
240 12.674668859 192.168.0.100 → 192.168.0.101 TCP 232 55 → 42428 [PSH, ACK] Seq=38769 Ack=481 Win=263 Len=176
```

Figure 11: Tshark capturing network data packets

The results conclude that IDS Snort, Honeypot Cowrie and Tshark packet analyzer were successfully configured and the proposed system was able to log ICMP interactions as well as a brute-force attack.

V. Conclusion

Unarguably, computer security is one of the biggest concerns of our generation. Tools such as intrusion detection systems, honeypots and packet analyzers when implemented correctly can very efficiently be utilized to counter cyber-attacks and ensure data integrity and security. Also, using inexpensive and easy to implement modules such as the raspberry pi can help in widespread usage of basic network monitoring tools [12].

In this paper, we have proposed a raspberry pi 3 equipped with Snort, an intrusion detection, Tshark, a packet analyzer and Cowrie, a full-fledged decoy honeypot server. After conducting our tests, the results proved that raspberry pi is a feasible solution to implement all the above three, without facing any memory or processing errors. Also, one of the main advantages of using the raspberry pi as a platform is the ease of implementation of such complex security tools, like honeypot.

The trends of cyber-attacks have just begun, and this project provides an efficient mechanism to combine various security tools to minimize attacks and maximize system security.

In conclusion, inexpensive network security solutions providing the ease of implementation have great potential in today's market. A low powered, portable network security device can prove to be an extremely efficient device for online security.

VI. References

[1] Sforzin, F. G. Marmol, M. Conti, and J.-M. Bohli, "RPiDS: Raspberry Pi IDS — A Fruitful Intrusion Detection System for IoT," 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016.

[2] Y. Turk, O. Demir, and S. Gören, "Real Time Wireless Packet Monitoring with Raspberry Pi Sniffer," Information Sciences and Systems 2014, pp. 185–192, 2014.

[3] A. K. Kyaw, Y. Chen, and J. Joseph, "Pi-IDS: evaluation of open-source intrusion detection systems on Raspberry Pi 2," 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec), 2015.

[4] H. Chaudhari, "Raspberry Pi Technology: A Review," International Journal of Innovative and Emerging Research in Engineering Volume 2, Issue 3, 2015.

[5] S. Mahajan, A.M Adagale and C. Sahare, "Intrusion Detection System Using Raspberry Pi Honeypot in Network Security," 2016 International Journal of Engineering Science and Computing Volume 6, Issue 3, 2016

[6] A. A. Ahmed and Y. W. Kit, "MICIE: A Model for Identifying and Collecting Intrusion Evidences," 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2016.

[7] T. Zitta, M. Neruda, and L. Vojtech, "The security of RFID readers with IDS/IPS solution using Raspberry Pi," 2017 18th International Carpathian Control Conference (ICCC), 2017.

[8] M. Cosar and S. Karasartova, "A firewall application on SOHO networks with Raspberry Pi and Snort," 2017 International Conference on Computer Science and Engineering (UBMK), 2017.

[9] D. Robinson and C. Kim, "A cyber-defensive industrial control system with redundancy and intrusion detection," 2017 North American Power Symposium (NAPS), 2017.

[10] A. Ranjan, S. Sinha, and S. Swain, "Signature Based Wireless Intrusion Detection Using Raspberry Pi," 2016 International Journal of Computer Engineering and Applications, Volume X, Special Issue, ICRTCST, 2016.

[11] C. Bousaba, T. Kazar, and W. Pizio, "Wireless Network Security Using Raspberry Pi," 2016 ASEE Annual Conference & Exposition Proceedings, 2016.

[12] A. H Villa, "Hands On Computer Security with Raspberry Pi," 2016 Journal of Computing Sciences in Colleges, Volume 31, Issue 6, 2016.

[13]Y Brar,T Choudhury et. al.,,An Advanced Security-A Two-Way Password Technique for Cloud Services,International Journal of Computer Science and Mobile Computing 3 (4), 4-15,2014