Unmanned Aerial Vehicle Attack Detection using Snort

Shahzad Mujeeb[1], Sunil Kumar Chowdhary[1], Abhishek Srivastava[1], Rana Majumdar[1] and Manoj Kumar[2]

[1]*Amity University Uttar Pradesh, Noida, India*

[2]*School of Computer Science, University of Petroleum and Energy Studies, Dehradun, India*

Keywords: Unmanned, Attack, Snort, Detection, Aerial, UAVs.

Abstract: In recent times, security issues relating to unmanned aerial vehicles (UAVs) and drones have anticipated a staid attention from research communities in various domains in the form of networking, communication, and civilian as well as in defence zone. It has its widespread functionality in the area of agriculture, commerce, and transportation, the use of unmanned aerial vehicles (UAVs)/ drones, is increasing. The ground control systems (GCS) are used to remotely monitor UAVs over the network. Since UAVs are vulnerable to security risk, they become the targets of various attacks such as GPS spoofing, jamming attack, network attacks and many other forms so to tackle with such issues the prime concern will be to identify these attacks followed by to prevent the UAVs or drones from UAV attacks. On contrary network-controlled UAVs however are equally vulnerable to threats like DOS attacks, GPS spoofing etc. In this work a network surveillance approach is projected for UAV attack detection system by means of Snort. Snort uses a set of guidelines and rules set by the user itself to help in identifying the malicious network behaviour and to locate packets that fit them and create user warnings with those rules. It is an open-source tool that records traffic analysis and packets in real time.

1 INTRODUCTION

An unmanned aerial vehicle UAVs commonly known as drones has become the most imperative device in today's technology era which exhibits its presence in various areas such as disaster monitoring, boarder surveillance, and relay communication. An automatic aircraft on board basically targets for reducing the deployment complexity, with low maintenance and acquirement cost. It aims to work with in conjunction with minimum user interaction in collaboration with UAV communication system ensuring robustness. The vehicular applications are used for connecting various multiple smart aerials over networks in order to meet the time critical missions. The only limitation to these unmanned aerial vehicles is they have limited data payload and energy. A remote-control mechanism is used to control the flights plan using software. Navigation refers to the process of accurately determining position, plans and path based on the given path.

Hence, drone navigation is the new research capacities that emphasis on developing system which help in measuring path plan and the position of various drones. It is the key to develop the next generation autonomous drones. These UAVs are functional almost ubiquitously in surveillance, transportation, monitoring, agriculture, farming, forestry, and environment protection. UAVs play an important and crucial role in military services where the UAVs are used for carrying small missiles and other weapons. These UAVs carry weapons; various missiles are being dropped at the intended positions controlled remotely from ground base stations and control stations. A large number of farmers may use these drones to check their fields, farms and crops as well in order to check irrigation systems, these UAVs can also be used in determining the areas of crops are impaired, damaged, broken so to make proper treatment accordingly. Retailers like Amazon, Flip-kart are using these drones to deliver various products. These UAVs are also used in film industries to take the beautiful cinematic shots nowadays; it is easily available for everyone to

Mujeeb, S., Chowdhary, S., Srivastava, A., Majumdar, R. and Kumar, M.
Unmanned Aerial Vehicle Attack Detection using Snort.
DOI: 10.5220/0010789700003167
In *Proceedings of the 1st International Conference on Innovation in Computer and Information Science (ICICIS 2021)*, pages 18-24
ISBN: 978-989-758-577-7

purchase and access at low prices to boot. Due to high mobility and flexibility more and more UAVs are being employed in civil applications, although it suffers from potential risk. Various threats that are associated with drones are the source of some risks; it may get collide with buildings, aircrafts or other objects, even it may get involved with potential terrorist acts as well.

Researchers show their concerns and choose this area as their research and presented their thoughts in various forms. Yuliya Averyanova, Lyudmila Blahaja in their works focuses on identifying UAV risks and vulnerabilities in order to better implement the risk-oriented approach of integrated UAVs into the airspace safely and improving the security of unmanned aerial systems. Nature, economic engineering, and vulnerabilities and threats specific to humans are quickly taken into account and certain potential approaches to reduce vulnerabilities and threats are also addressed in the paper.

Menaka Pushpa Arthur talks about various possible cyber and physical risks that could emerge from the use of UAVs, and then investigate multiple methods of identifying, monitoring, and interdicting hostile drones by utilizing techniques that focus on UAV-emitted ambient radio frequency signals, radars, acoustic sensors, and UAV-detection computer vision techniques. Yuliya Averyanova, Lyudmila Blahaja showed their concers about durability of such vehicles. H. Shakhatreh, A.H. Sawalmeh, A.I. Al-Fuqaha, Z. Dou, E.K. Almaita, I.M. Khalil emphasis on key research challenges in this area that need to be addressed properly. A.A. Zhilenkov, I.R. Epifantsev talks about trajectories planning in navigation system. H. Sedjelmaci, S. M. Senouci and N. Ansari depicted that UAVs or drones have been vulnerable to multiple malware attacks such as the jamming attack since FANET.

The paper proposes a security framework for FANET for Federated Learning-based on-device jamming attack detection. It concludes that GPS Jamming and Spoofing concentrate on UAV-related research to address cybersecurity risks but avoids assaults on the stream of controls and data communications. L. Xiao, C. Xie, M. Min and W. Zhuang [8] discussed the practicality of using Identity Based Encryption in the UAV resource restriction network. A major architecture challenge when encryption is applied is the space limitation existence of such WiFi-based UAV networks as elaborated by Park, K. J., Kim, J., Lim, H., & Eun, Y. It also discusses the practicality and performance of IBE Identity Based Encryption in the UAV network and thus provides an important wireless UAV network resource limited security system.

From literature review it has been observed that most of the failures are due to:

- Technical breakdown.
- Human factor.
- Adverse weather.
- Other factors.

However, in some intricate surroundings, UAV cannot sense the environment parameters due to limited communication and traditional sensor perception capabilities. Despite many efforts to overcome these weaknesses, it is essential to develop more efficient and effective method in order to perform more stability, predictability, and security. Therefore, high performance independent navigation is of great importance to develop the application of UAV is of great importance. The control of each drone falls on pilot to use visual tracking to determine position and orientation. More advanced drones use global positioning system (GPS) receivers to play a significant role, that is, navigation and control loop. Some smart features include drone memorization to track the position track. The trajectory of the drone can be predetermined to establish GPS waypoints. When this function is executed, the drone will use autopilot to follow this path.

There exist various forms of UAV attacks; the initial stages of UAV attack start with affecting the physical configuration and the loss of mobility which is also known as manoeuvrability. It is very much difficult to detect, and to prevent or countermeasure which includes the proper capturing of vulnerabilities. On the basis of certain factors, UAV attack can be categorized into four major parts which are named as, UAV freezing, waypoint alterations, enforced collision, and UAV hijacking.

These various attacks are as follows:

- UAV freezing: This attack starts with the failure of node which is caused due to modification in physical configuration of unmanned armed vehicles which leads to loss in mobility of UAV. These mobility losses result in network failures. Intrusion, signal jamming, and session hijacking are the main cause to this attack.

- Waypoint alterations: Another major threat to fully functional UAVs is waypoint modifications. This attack leads to overlapping of mobility patterns which in turn results in enforced collision. This is a very fatal attack, also very difficult to detect, identify and trace their effect.

- Enforced clustering: This attack is opposite to waypoint alterations, in this attack UAV adjust itself forming sub-clusters, and also create the own sub-network that works and operates opposite to the existing networks. It can be used in getting various significant details regarding configurations and patterns of existing UAVs.
- UAV hijacking: This attack is capturing of UAVs from remote location using connective technology. Since UAVs are controlled remotely from ground base station or control stations so it becomes easy for the attacker to hack in between and make the UAV work as per their requirements and intentions. UAVs operate initially under the control of the third party to override the instructions if required.

The attacks are created based on the identified vulnerabilities found in a system, in this case UAV vulnerabilities which result in various number of attacks such as attack to physical configuration, waypoint alteration, UAV high-jacking, UAV network attack such as DOS attack, GPS Spoofing and so on. These attacks can affect a single node or a single link between two nodes or it can also affect the whole networks which could lead to huge loss.

These attacks can split up into three phases which are named as, identification phase, creating vulnerabilities/ session break, and attack phase. The details of each phase are as follows:

- Identification phase: It is the first and the initialisation phase starting with the network to operate the entire network to fetch the data in network using certain existing approaches like session hijacking and so on.
- Session's break/creating vulnerabilities: Since all the networks are prone to network attacks to acquire a session to launch any one of the defined attacks. In this second phase, a new set of codes can be used to launch cyber-attack.
- Attack Phase: This is the final phase after the attack is launched; the cyber-attack starts with compromising the whole network to revoke the session. The main objective of a network controller is to prevent a network from undergoing the state of cyber-attack during network mission because once it is initialised it becomes very difficult to detect and prevent it.

The term vulnerability refers to the unreliability and insecurity; but in the context of UAVs vulnerabilities can be defined as the weakness in a system that could lead to the misuse of that system or loss such as system damage. In order to comprehend and systemize these, which can be revealed while UAV operation phase, a basic general taxonomy was developed. It allows in tracking certain possible vulnerabilities in UAV life cycle though its stages, as well define their purposes, ways and subjects and so on. As it is shown in figure below, these vulnerabilities can occur at four stages of UAV life cycle: development and design phase, manufacturing phase, operation phase and maintenance phase. These vulnerabilities that can occur at each stage of its lifecycle can further be divided into incidental or intentionally which is created for occasional or specific purpose.

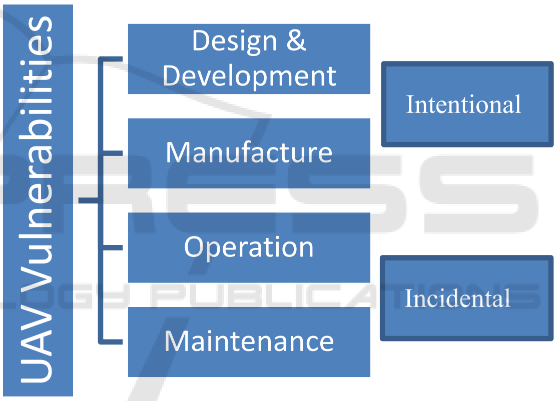
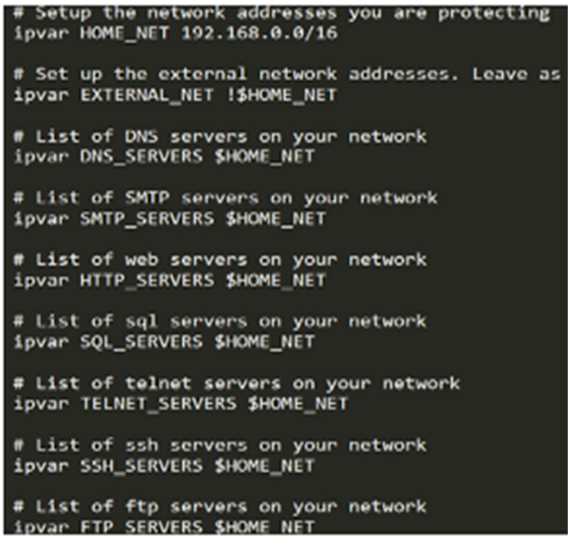


Figure 1: Classification of Vulnerabilities Depending on the Phase of UAV Lifecycle

2 METHODOLOGY

Snort is the world's leading Open-Source packet analysis software. It was created in 1998 by Martin Roesch, founder and former CTO of Sourcefire. Snort is now developed by Cisco, which purchased Sourcefire in 2013. Snort uses a set of guidelines to help identify malicious network behaviour and to locate packets that fit them and create user warnings with those rules. It is used for performing real time traffic analysis. A victim machine can use Ubuntu or Windows operating system in which snort is configured and using kali Linux an attack/scan is performed. A configuration file in snort contains all details about servers running in your network as shown in the figure 2.



```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.0/16

# Set up the external network addresses. Leave as
ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET
```

Figure 2: List of Servers in Snort.Conf.

Snort (Snort, 2018) can be configured in three main modes: sniffer, packet logger, and detection mode.

- Sniffer Mode: This phase consists of several processes such as reading of network packets and displaying in the console.
- Packet Logger Mode: In this phase packet logger mode is enabled and certain programs are present to log packets to the disk.
- Detection System Mode: This is the last phase in snort in which the detection mode is on, monitoring of network packets and analysing is done based on the rule set given by the user. After detection the defined actions are performed.

Snort is based on libpcap, a method that is commonly used in TCP / IP traffic sniffers and analysers (for library packet capture). Snort detects attack techniques, including denial of service, buffer overload, CGI attacks, stealth port checks, and SMB probes, by protocol analysis and content scanning and matching. Snort sends a real-time warning to syslog, a separate 'alerts' file, or to a pop-up window when unusual activity is observed.

Snort rules are used to identify the malicious packet incoming in the network then using these rules user define what he/she wants to do with malicious packet identified i.e to discard it, dropt it, log it .Snort rules are basically used to handle the packets.

In snort one can even define his/her own customise rules.
alert icmp $ETERNAL_NET any -> $HOME_NET any (msg:"Shahzad message"; sid:314;rev:5;)

2.1 Process Adopted

Since IP address is a specific address for each device which is used to transfer data or packets from one network to the other network over the internet. There is a message, data, source, destination address, and much more in-packet. Three IP protocols for suspicious behaviour are provided by Snort:

- The Transmission Control Protocol (TCP) links and transfers data between two separate hosts. HTTP, SMTP, and FTP are examples.
- User Datagram Protocol (UDP): Internet broadcasts messages. Examples include traffic via DNS.
- ICMP (Internet Control Message Protocol): Transfers Windows error messages to the network. Ping and Traceroute are examples.

A victim machine can use Ubuntu or Windows operating system in which snort is configured and using kali Linux an attack/scan is performed. Snort rules are used to identify the malicious packet incoming in the network then using these rules user define what he/she wants to do with malicious packet identified i.e., to discard it, drop it, log it. Snort rules are basically used to handle the packets. In snort one can even define his/her own customise rules.

I. It starts with downloading and installing Snort. Snort uses the common libpcap library, winpcap (for Windows), which is the same library used by tcpdump to sniff packets.

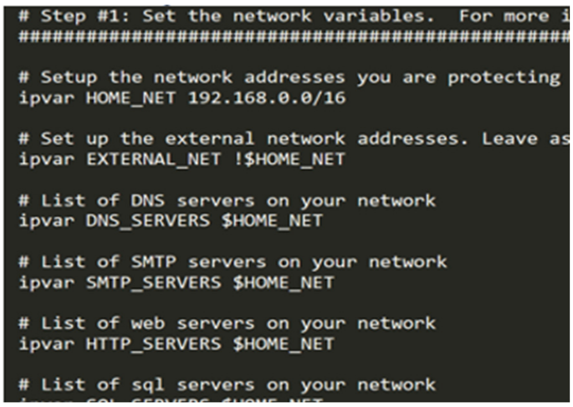
A single line must include the Snort rules. The snort rule parser does not accommodate multi-line rules unless you use a multi-line character \. It is normally included in the configuration file for snort.conf.

Two rational pieces come with this:

- Rule header: Defines the behaviour of the rules, such as alerts, register, transfer, trigger, dynamic and CDIR block.
- Rule options: Define the warning messages of the regulation.

II. The first variable $HOME_NET is changed to machine IP address e.g. 192.168.50.8, while the $EXTERNAL_ NET is left as it is except for the $HOME_NET.

III. The next step is change in snort rule options, so to the RULE_PATH, replace ../rules with c:\Snort\rules and replace ../ so_rules with c:\Snort\ so_rules. At last, replace../ preproc_ rules with c:\Snort\ preproc_ rules.



```
# Step #1: Set the network variables.  For more i
###################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.0/16

# Set up the external network addresses. Leave as
ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
```

Figure 3: HOME_NET IN SNORT.CONF FILE.

IV. Create a directory in snort rule of whitelist directory same as blacklist directory.

V. Change the configuration of login directory in a way #config logdir: to config logdir:c:\Snort\log. It helps in presenting the output in a particular location.

VI. Now configure the dynamic loaded libraries. At path to dynamic preprocessor libraries, replace usr/local/lib/snort_dynamicpreprocessor with your dynamic preprocessor, which is C:\Snort\lib\snort_dynamic -preprocessor. Similarly, replace usr/ local/lib/snort_dynamicengine/libsf_engine.so with the base pre-processor engine, which is C:\Snort\lib\snort_ dynamicengine\sf_engine.dll.

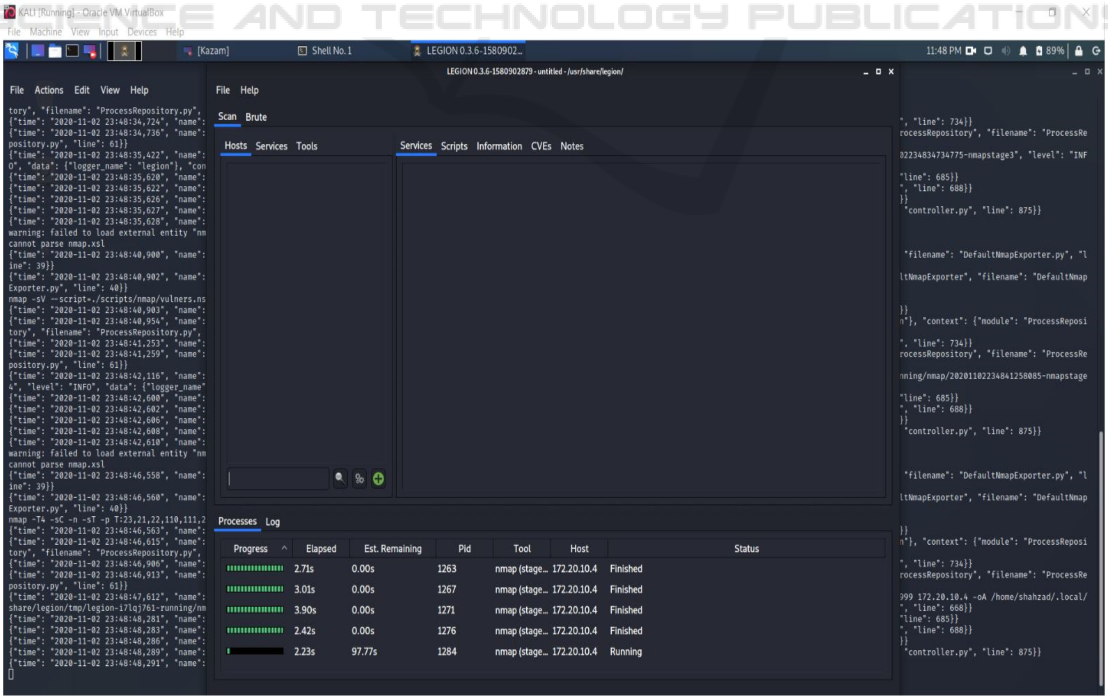
VII. Once the snort is configured with the given set of rules, it starts capturing packets as shown in the figure 4, which include the alert packet, packet type, and the IP-address and the priority. It starts monitoring against the cyber-attacks.

Snort –A console – c/etc/snort/snort.conf.

VIII. As the packet start reaching the victim machine, it is captured by snort with alert message as well with the IP address of the attacker machine.

IX. Now the snort rules come into play, as the malicious packets has been identified, now snort rules either discard the packets or log the packets or to reject the packet based on the user defined rules. The user can also define customised rules as per requirement.

alert icmp $ETERNAL_NET any -> $HOME_NET any (msg:" Shahzad message"; sid:314; rev:5;).


KALI [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[Kazam]
Shell No. 1
LEGION 0.3.6-1580902...
11:48 PM
89%
LEGION 0.3.6-1580902879 - untitled - /usr/share/legion/
File Actions Edit View Help
File Help
Scan Brute
Hosts Services Tools
Services Scripts Information CVEs Notes
Processes Log

```
tory", "filename": "ProcessRepository.py",
{"time": "2020-11-02 23:48:34,724", "name":
{"time": "2020-11-02 23:48:34,736", "name":
pository.py", "line": 61}}
{"time": "2020-11-02 23:48:35,422", "name":
0", "data": {"logger_name": "legion"}, "con
{"time": "2020-11-02 23:48:35,620", "name":
{"time": "2020-11-02 23:48:35,622", "name":
{"time": "2020-11-02 23:48:35,626", "name":
{"time": "2020-11-02 23:48:35,627", "name":
{"time": "2020-11-02 23:48:35,628", "name":
warning: failed to load external entity "nm
cannot parse nmap.xsl
{"time": "2020-11-02 23:48:40,900", "name":
ine": 39}}
{"time": "2020-11-02 23:48:40,902", "name":
Exporter.py", "line": 40}}
nmap -sV --script=./scripts/nmap/vulners.ns
{"time": "2020-11-02 23:48:40,903", "name":
{"time": "2020-11-02 23:48:40,954", "name":
tory", "filename": "ProcessRepository.py",
{"time": "2020-11-02 23:48:41,253", "name":
{"time": "2020-11-02 23:48:41,259", "name":
pository.py", "line": 61}}
{"time": "2020-11-02 23:48:42,116", "name":
4", "level": "INFO", "data": {"logger_name"
{"time": "2020-11-02 23:48:42,600", "name":
{"time": "2020-11-02 23:48:42,602", "name":
{"time": "2020-11-02 23:48:42,606", "name":
{"time": "2020-11-02 23:48:42,608", "name":
{"time": "2020-11-02 23:48:42,610", "name":
warning: failed to load external entity "nm
cannot parse nmap.xsl
{"time": "2020-11-02 23:48:46,558", "name":
ine": 39}}
{"time": "2020-11-02 23:48:46,560", "name":
Exporter.py", "line": 40}}
nmap -T4 -sC -n -sT -p T:23,21,22,110,111,2
{"time": "2020-11-02 23:48:46,563", "name":
{"time": "2020-11-02 23:48:46,615", "name":
tory", "filename": "ProcessRepository.py",
{"time": "2020-11-02 23:48:46,906", "name":
{"time": "2020-11-02 23:48:46,913", "name":
pository.py", "line": 61}}
{"time": "2020-11-02 23:48:47,612", "name":
share/legion/tmp/legion-i7lqj761-running/nm
{"time": "2020-11-02 23:48:48,281", "name":
{"time": "2020-11-02 23:48:48,283", "name":
{"time": "2020-11-02 23:48:48,286", "name":
{"time": "2020-11-02 23:48:48,289", "name":
{"time": "2020-11-02 23:48:48,291", "name":
```

Progress	Elapsed	Est. Remaining	Pid	Tool	Host	Status
▮▮▮▮▮▮▮▮▮▮	2.71s	0.00s	1263	nmap (stage...	172.20.10.4	Finished
▮▮▮▮▮▮▮▮▮▮	3.01s	0.00s	1267	nmap (stage...	172.20.10.4	Finished
▮▮▮▮▮▮▮▮▮▮	3.90s	0.00s	1271	nmap (stage...	172.20.10.4	Finished
▮▮▮▮▮▮▮▮▮▮	2.42s	0.00s	1276	nmap (stage...	172.20.10.4	Finished
▮	2.23s	97.77s	1284	nmap (stage...	172.20.10.4	Running

```
", "line": 734}}
rocessRepository", "filename": "ProcessRe

02234834734775-nmapstage3", "level": "INF

"line": 685}}
", "line": 688}}
}}
 "controller.py", "line": 875}}


 "filename": "DefaultNmapExporter.py", "l

ltNmapExporter", "filename": "DefaultNmap


}}
n"}, "context": {"module": "ProcessReposi

", "line": 734}}
rocessRepository", "filename": "ProcessRe

nning/nmap/20201102234841258085-nmapstage

"line": 685}}
", "line": 688}}
}}
 "controller.py", "line": 875}}


 "filename": "DefaultNmapExporter.py", "l

ltNmapExporter", "filename": "DefaultNmap


}}
n"}, "context": {"module": "ProcessReposi

", "line": 734}}
rocessRepository", "filename": "ProcessRe

999 172.20.10.4 -oA /home/shahzad/.local/
", "line": 668}}
"line": 685}}
", "line": 688}}
}}
 "controller.py", "line": 875}}
```

Figure 4: Network Mapping Through Legion as an Attacking Machine.

X. The next step is the attacking phase in which kali linux with any penetration testing tool like SPARTA or LEGION in which the IP address of the attacked machine having snort is input.

3 RESULTS AND DISCUSSIONS

Snort laws must be written so that they correctly define any of the following events:

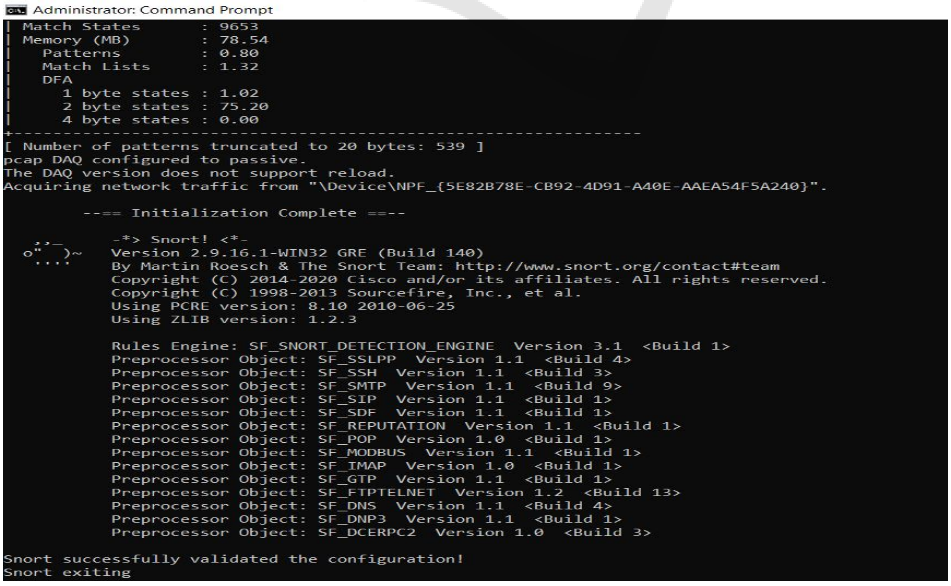
- Conditions in which a user assumes that the network packet(s) is not the same as normal, or that the packet identity is not authentic.
- Any infringement of the company's security policies that could endanger the security of the network of the company and other sensitive details.
- Both well-known and famous attempts in the company's network to manipulate the vulnerabilities.

Based on the extent of the intrusion, the laws defined by the system should be reasonably compatible in order to respond promptly and take the required corrective steps. The rules are not tested by Snort in the order that they appear in the file of snort rules. The sequence, by implication, is:

- Alert rules: Use the alert system to create an alert.
- Log rules: After the alert is produced, the packet is then logged.
- Pass rules: It refuses and drops the packet.

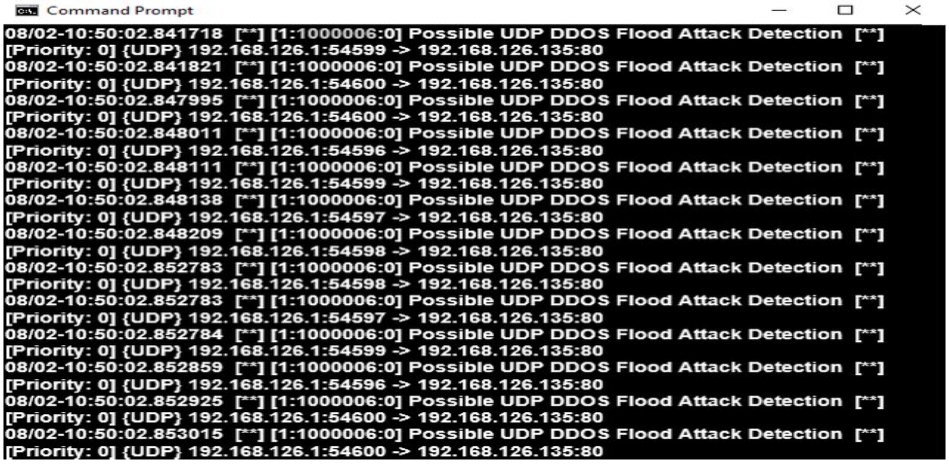
4 CONCLUSIONS

Security is obligation as the UAVs are used in every aspect from the military operations to day-to-day life, being controlled remotely from ground base station and any intrusion to its network can cause huge damage which could be uncountable. Here the authors recommended the process of UAV attacks identification implementing network management and the packet analysis tool. The proposed method using snort helped in detecting DOS attack, the GPS spoofing attack as shown in figure 5 & 6. The method of detection has the benefit of concurrently identifying multiple threats using simulation. The above method can be used and implemented for a short overview and analysis of vulnerabilities and threats that can become the source of the risks for UAVs.



```
| Match States       : 9653
| Memory (MB)        : 78.54
|   Patterns         : 0.80
|   Match Lists      : 1.32
|   DFA
|     1 byte states : 1.02
|     2 byte states : 75.20
|     4 byte states : 0.00
+-----------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 539 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{5E82B78E-CB92-4D91-A40E-AAEA54F5A240}".

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.16.1-WIN32 GRE (Build 140)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2020 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>

Snort successfully validated the configuration!
Snort exiting
```

Figure 5: Snort testing successful.



```
08/02-10:50:02.841718  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54599 -> 192.168.126.135:80
08/02-10:50:02.841821  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54600 -> 192.168.126.135:80
08/02-10:50:02.847995  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54600 -> 192.168.126.135:80
08/02-10:50:02.848011  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54596 -> 192.168.126.135:80
08/02-10:50:02.848111  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54599 -> 192.168.126.135:80
08/02-10:50:02.848138  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54597 -> 192.168.126.135:80
08/02-10:50:02.848209  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54598 -> 192.168.126.135:80
08/02-10:50:02.852783  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54598 -> 192.168.126.135:80
08/02-10:50:02.852783  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54597 -> 192.168.126.135:80
08/02-10:50:02.852784  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54599 -> 192.168.126.135:80
08/02-10:50:02.852859  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54596 -> 192.168.126.135:80
08/02-10:50:02.852925  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54600 -> 192.168.126.135:80
08/02-10:50:02.853015  [**] [1:1000006:0] Possible UDP DDOS Flood Attack Detection  [**]
[Priority: 0] {UDP} 192.168.126.1:54600 -> 192.168.126.135:80
```

Figure 6: Snort Detecting DDOS Attack.

REFERENCES

D. Muniraj and M. Farhood. Detection and mitigation of actuator attacks on small, unmanned aircraft systems. Control Engineering Practice, vol. 83, pp. 188–202, Feb 2019.

Menaka Pushpa Arthur. Detecting Signal Spoofing and Jamming Attacks in UAV Networks using a Lightweight IDS. 978-1-7281-1374-6/19/$31.00 ,2019 IEEE.

Yuliya Averyanova, Lyudmila Blahaja. A Study on Unmanned Aerial System Vulnerabilities for Durability Enhancement. 978-1-7281-2592-3/19/$31.00 2019 IEEE.

H. Shakhatreh, A.H. Sawalmeh, A.I. Al-Fuqaha, Z. Dou, E.K. Almaita, I.M. Khalil, et al., Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges, IEEE Access 7 (2019) 48572–48634.

A.A. Zhilenkov, I.R. Epifantsev, Problems of a trajectory planning in autonomous navigation systems based on technical vision and AI, in IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, IEEE, 2018.

H. Sedjelmaci, S. M. Senouci and N. Ansari. A Hierarchical Detection and Response System to Enhance Security Against Lethal Cyber-Attacks in UAV Networks. in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 9, pp. 1594-1606, Sept. 2018.

C. Li, Y. Xu, J. Xia and J. Zhao. Protecting Secure Communication Under UAV Smart Attack With Imperfect Channel Estimation in IEEE Access, vol. 6, pp. 76395-76401, 2018.

L. Xiao, C. Xie, M. Min and W. Zhuang. User-Centric View of Unmanned Aerial Vehicle Transmission against Smart Attacks. in IEEE Transactions on Vehicular Technology, vol. 67, no. 4, pp. 3420-3430, April 2018.

Park, K. J., Kim, J., Lim, H., & Eun, Y.Robust Path Diversity for Network Quality of Service in Cyber-Physical Systems. IEEE Trans. Industrial Informatics, 10(4), 2204-2215.

Snort. Accessed on August. 27, 2018. [online] Available: https://www.snort.org/

K. Huang and H. Wang. Combating the Control Signal Spoofing Attack in UAV Systems. in IEEE Transactions on Vehicular Technology, vol. 67, no. 8, pp. 7769-7773, Aug. 2018.

Bakkiam David Deebaka, Fadi Al-Turjmanb. Aerial and underwater drone communication: potentials and vulnerabilities.TJCME(2018).