Security Network Lab (MITM & Honeypot)

César Rodríguez Morales
Master in Computer Science
Advisor: Dr. Jeffrey Duffany
Electrical & Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *− Many people on the IT environment knows that internet can be a dark and dangerous place; featuring viruses and cyber-attacks. But the rest of the people across the internet not think that are really exposed to be attacked. This project aims to uncover some of these threats and reveal just how vulnerable the internet can be. This project involves the implementation of a honeypot (a device designed to attack and observe the cyber attackers behavior) and to analyze cyber-attacks to see what is going on in the dark underworld of the internet. This project will tried to explains the process involved in building a honeypot in Linux along with the results produced from that honeypot. The honeypot logs will be analyses to gain an understanding into how cyber-attackers operate and to determinate the most common attacks to a home network and also with integrate the one of the most used cyberattack on internal networks.*

Key Terms *− Cyber-Attacks, HoneyPot, Internet, MITM, Security, Threats, Vulnerabilities.*

INTRODUCTION

To introduce this project is necessary identify that the main focus for this project is the behavior and analysis of threat through the building a honeypot to research cyber-attack techniques and will also use the famous "man in the middle" to see the importance of use a secure WIFI connection. This section provides an introduction to some common internet threats, what a honeypot is and why honeypots are useful at detecting cyber threats, what the SSH protocol is and what the aims for this project are. Today we can say that the internet can be a dangerous place due to the many threats that exist within it. The people most of the time we may be completely unaware of these threats since they are hidden and might not be immediately obvious. The good and bad of the internet is that is made up of a globally distributed network; it is not run by one single organization or company. Therefore the internet has no central management or governing body. This means that the internet has no global laws. That means that may be illegal in one country may not be illegal in another country. Another issues facing the internet is anonymity, so many users of the internet believe that their actions online cannot be traced since they are "hidden" behind a computer. This person also use VPN services to perform the attacks and thinks that they are untouchable. These reasons may lead some internet users to carry out actions which may be considered unethical or illegal only in some countries, such as cyber-attacks. As a result: the internet is littered with many cyber threats and cyber-attacks which are carried out by these unethical internet users which we call attackers. The other important part of this project is the use of the cyber-attack named man-in-the-middle (MITM) is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other.

BACKGROUND

The term Cybersecurity includes controlling physical access to the hardware, as well as protecting against harm that may come via network access, data and code injection. From here on viruses went, well, viral and dominated the headlines. The Melissa and ILOVEYOU viruses infected tens of millions of PCs, causing email systems around the globe to fail, all with little strategic objective or clear financial motivation. These threats led to the development of antivirus technology in order to spot the signature of the virus and prevent it from executing. Equally as important, these threats also played a huge role in driving the awareness of computer users of the risks of reading emails from untrusted sources and opening their attachments. This realization was not lost on companies, as it became clear that if viruses were to spread from corporate email accounts, questions about the security and integrity of the company could be brought into the public eye.

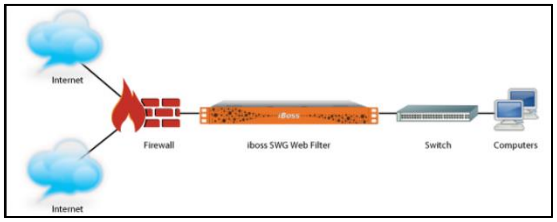


Figure 1
Basic Security Network Diagram

In 1989, Robert Morris created what is now widely acknowledged as the first computer worm. This self-propagating virus spread so aggressively and rapidly that it succeeded in closing down much of the internet. While other subsequent attacks have gained far more notoriety, the Morris worm was a landmark incident in that it was the first widespread instance of a denial-of-service (DoS) attack. Due to the infancy of the internet at the time, the impact was nowhere near as devastating as it would be today. However, it laid the groundwork for the kinds of security issues that we've seen ever since. The Morris worm and the early nuisance attacks that followed were early instances of having to deal with, and respond to, cyber-security attacks. They ultimately led to the security industry as we know it – including the establishment of CERTs (Computer Emergency Response Teams) as a central point for coordinating responses to these kinds of emergencies. The initial reaction from the industry followed the old adage ‘prevention is better than a cure’, giving rise to what has become a litany of preventative and detective security products.

In the other hand we have the case of the MITM the background history or information is very little. But is documented by Larsen, Gerald H. “Software: A Qualitative Assessment, or The Man in the Middle Speaks Back.” Datamation 19 (November 1973) the first book dedicate to this attack. “Such an attack might be utilized when it is not possible to take advantage of other weaknesses in an encryption system that would make the task easier” [1].

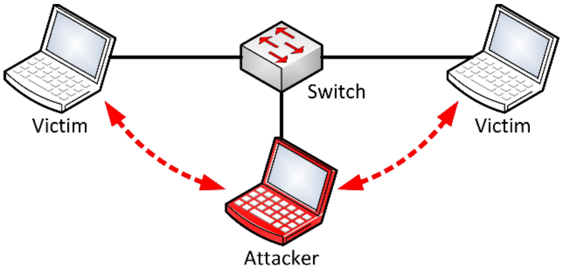


Figure 2
Man in the Middle Diagram

PROBLEM

The importance of this project is to be able to see more clearly how vulnerable we are and we dangerous can be the public WIFI. This creates a big problem since many of the users who surf through the internet are not properly protected. In certain cases they only have a free antivirus program that does not include all the necessary tools. And if we talk about more complex terms like a firewall, the amount increases alarmingly. This is not due to its complexity because there are many security solutions pre-configured and others very easy to configure with little or no knowledge of systems. This is not here, the risk is even greater when employees with devices from their companies and companies connect at home to perform tasks outside a controlled environment as they usually are the places of work. A lot of information can be exposed and the employee does not even know it.

METHODOLOGY

The central idea of this research project is to create a security laboratory that is composed of 5 main elements: (HoneyPot, IDS, Internal Firewall, External Firewall, Switch).

The honeypot means, "jar of honey". It is a tool that is used almost exclusively in the field of computer security. Its function is based on attracting and analyzing attacks made by bots or hackers. Its objective is to attract attackers to see their attack patterns, generate dictionaries to collect what words they use in attacks, know the enemy and their profile. We can say that a honeypot is a great ally to defend your network, although the concept is to attract attacks, the best tactic to defend is to know how the enemy acts and what better to put a bait which will not pose a risk to our network, and it will help us to know how the attack affects and in what way [2].

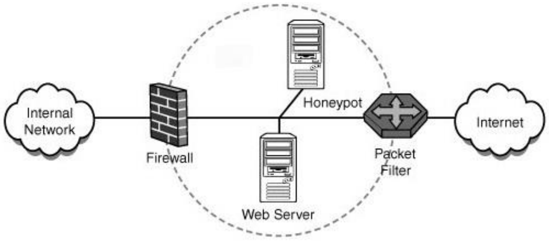


Figure 3
Honeypot Network Architecture

In this image we see the components of the Honeypot Network Architecture. This is not the only implemented design, the honeypot can be used on the external facing also. In addition, on this project the honeypot will be located on the DMZ zone in order to protect the internal network of infections.

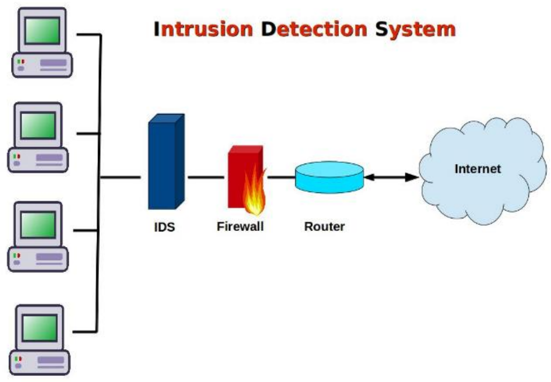


Figure 4
IDS Diagram

IDS (Intrusion Detection System), is a program to detect unauthorized access to a computer or a network. The IDS usually has virtual sensors with which the IDS core can obtain external data (usually on network traffic). The IDS detects, thanks to these sensors, anomalies that may indicate the presence of attacks and false alarms. The operation of these tools is based on the detailed analysis of network traffic, which upon entering the analyzer is compared with signatures of known attacks, or suspicious behavior, such as port scanning, malformed packets, etc. The IDS not only analyzes what type of traffic it is, but also reviews the content and its behavior. Normally this tool is integrated with a firewall. The intrusion detector is unable to stop the attacks on its own, except those that work together in a gateway device with firewall functionality, making it a very powerful tool since it joins IDS intelligence and blocking power of the firewall, being the point where the packets must necessarily pass and can be blocked before penetrating the network. The IDS usually have a database of "signatures" of known attacks.

Firewall is a software program that prevents unauthorized access to or from a private network. Firewalls are tools that can be used to enhance the security of computers connected to a network, such as LAN or the Internet. They are an integral part of a comprehensive security framework for your network. A firewall absolutely isolates your computer from the Internet using a "wall of code" that inspects each individual "packet" of data as it arrives at either side of the firewall — inbound to or outbound from your computer — to determine whether it should be allowed to pass or be blocked. Firewalls have the ability to further enhance security by enabling granular control over what types of system functions and processes have access to networking resources. These firewalls can use various types of signatures and host conditions to allow or deny traffic. Although they sound complex, firewalls are relatively easy to install, setup and operate [3].

Most people think that a firewall is a device that is installed on the network, and it controls the traffic that passes through the network segment. However, you can have a host-based firewalls. This can be executed on the systems themselves, such as with ICF (Internet Connection Firewall). Basically, the work of both the firewalls is the same: to stop intrusion and provide a strong method of access control policy. In simple definition, firewalls are nothing but a system that safeguards your computer; access control policy enforcement points.

After explaining more thoroughly the three most important terms of this project (IDS, Firewall and Honeypot). It only remains to identify how its implementation will proceed. Firstly, the configuration of the external firewall will be done since this is the first external protection layer. From this point of the perimeter, the DMZ interface is configured, where two of the main elements (IDS and Honeypot) will be located. Then we will proceed to configure the internal area where we will have internal devices such as computers and cell phones that we will keep away from the test, since it represents a risk of infection. Then we will work with the configuration of the UTM where the filtering of the network resides and the inspection of packages through the IPS. With this completed we are ready to start working with the configuration of the honeypot. Currently there are many types of honeypot focused on a particular service such as Kippo Honeypot or with multiple vulnerabilities like the one we will be using. After completing the installation and programming of the honeypot we will be ready to see its behavior through the IDS that for this laboratory we will use one of the best known in the industry "SNORT". With the IDS implemented, it only remains to place the computer in the middle of the DMZ transmission to collect the data [4].

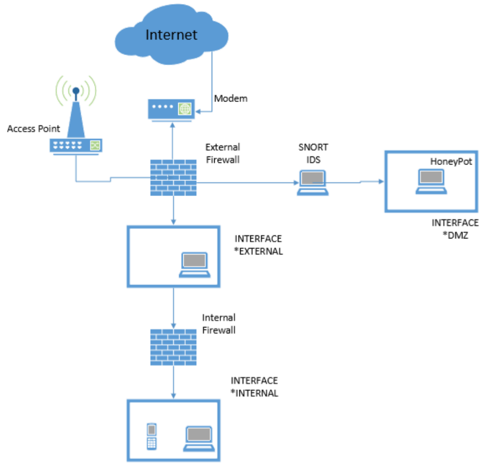


Figure 5
Network Diagram

The diagram that was created was strategically designed to apply with both tests. For this reason we use the integration of an access point in which the port mirror was made. This has 2 advantages, the first is that it allows you to connect multiple devices without the need for cables, and the second is the ease with which users can be deceived. To carry out tests with MITM there are many tools on platforms such as KALI LINUX that allow us to obtain data without users perceiving it. For project purposes I will use the wireshark tool for the ability to capture traffic and for the ease with which it can be obtained by any user as it is completely free. To perform these tests I will use the NMAP tool to verify its behavior through the port mirror.

RESULTS AND DISCUSSION

In this stage of results and discussion, we must undoubtedly touch on very important points such as stage of implementation and how during the realization of the project we obtained information and if the same complied with the initial expectations and what would be the steps to follow in the future.

A-Hardware Connection and Setup

To begin the first step was the implementation of the firewall since it is the heart of the project. This is because thanks to it, it was possible to separate the internal network from the external network which was vulnerable and without protection. The necessary policies were created to allow traffic and the interfaces in which the DMZ was created and in which the TRUNK port was connected in order to connect the access point and the port mirror.

Figure 6
Firewall (Front View)

At the level of Switch, I proceeded to enable the port where the firewall would connect and likewise was configured to replicate the traffic generated by port 1 to port 8 where it would be our monitoring computer. At this point is where it is possible to perform the MITM attack since there is a promiscuous interface showing all the traffic which can be analyzed in a malicious way.

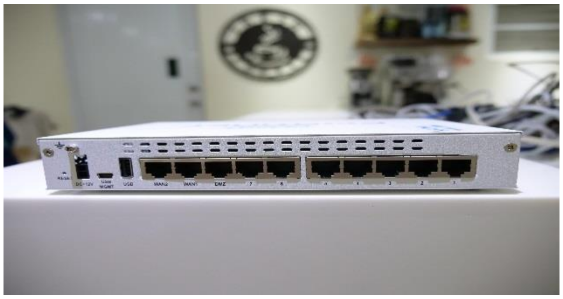


Figure 7
Firewall (Rear View)

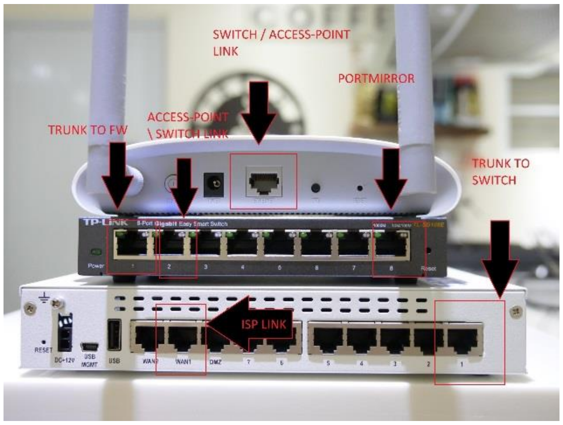


Figure 8
Network Devices

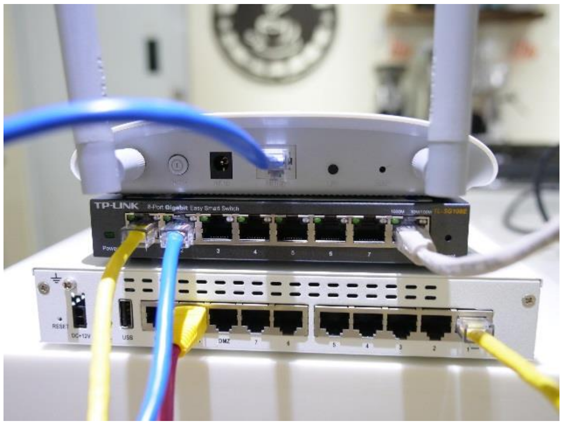
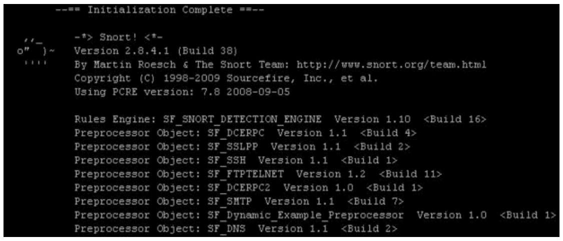


Figure 9
Network Devices Connected

After working with the physical part, we proceeded to work with the implementation of what would be the IDS and the other tools used for monitoring. For this, virtual machines were created in which was installed the operating system of KALI LINUX it has the advantage of the tools that It already has pre-installed which are very useful. In the other virtual machine, the distribution of Security Onion which like Kali is a Linux distribution but is focused on monitoring tools like SNORT and SURICATA. For the MITM stage, the wireshark tool was installed with which the traffic capture was analyzed.

Intrusion Detection System Setup

As an initial part of this stage, SNORT and its respective signatures were installed. It is important to know that the complexity of the SNORT increases according to your addictions and configuration.



```
        --== Initialization Complete ==--

  ,,_     -*> Snort! <*-
 o"  )~   Version 2.8.4.1 (Build 38)
  ''''    By Martin Roesch & The Snort Team: http://www.snort.org/team.html
          Copyright (C) 1998-2009 Sourcefire, Inc., et al.
          Using PCRE version: 7.8 2008-09-05

          Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.10  <Build 16>
          Preprocessor Object: SF_DCERPC  Version 1.1  <Build 4>
          Preprocessor Object: SF_SSLPP  Version 1.1  <Build 2>
          Preprocessor Object: SF_SSH  Version 1.1  <Build 1>
          Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 11>
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 1>
          Preprocessor Object: SF_SMTP  Version 1.1  <Build 7>
          Preprocessor Object: SF_Dynamic_Example_Preprocessor  Version 1.0  <Build 1>
          Preprocessor Object: SF_DNS  Version 1.1  <Build 2>
```

Figure 10
Snort Installation

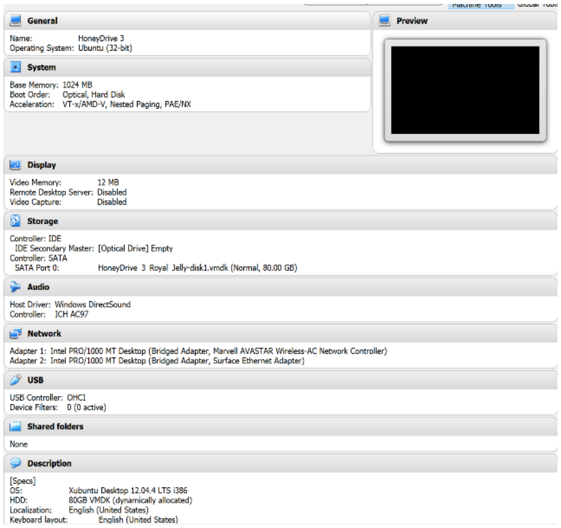
The installation of SNORT was divided into 4 stages.

a) Network Card Configuration
b) Snort Pre-Requisites:
    i) pcap (libpcap-dev)
    ii) PCRE (libpcre3-dev)
    iii) Libdnet (libdumbnet-dev)
    iv) DAQ
c) Snort Installation
d) Snort Configuration
e) Snorby Snort Events

Honeypot Setup

In this third stage, we already have the two most important stages completed. And we need to establish a vulnerable target in which we can expose the internet to analyze (source IP, destination IP and protocols).

Having all the tools already configured, we proceeded to the configuration of the last virtual machine in which a Linux distribution called HONEY DRIVE was installed, which contains tools of honeypots.



```
General
Name:              HoneyDrive 3
Operating System:  Ubuntu (32-bit)

System
Base Memory:  1024 MB
Boot Order:   Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Preview

Display
Video Memory:           12 MB
Remote Desktop Server:  Disabled
Video Capture:          Disabled

Storage
Controller: IDE
  IDE Secondary Master:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:           HoneyDrive 3 Royal Jelly-disk1.vmdk (Normal, 80.00 GB)

Audio
Host Driver:  Windows DirectSound
Controller:   ICH AC97

Network
Adapter 1:  Intel PRO/1000 MT Desktop (Bridged Adapter, Marvell AVASTAR Wireless-AC Network Controller)
Adapter 2:  Intel PRO/1000 MT Desktop (Bridged Adapter, Surface Ethernet Adapter)

USB
USB Controller:  OHCI
Device Filters:  0 (0 active)

Shared folders
None

Description
[Specs]
OS:             Xubuntu Desktop 12.04.4 LTS i386
HDD:            80GB VMDK (dynamically allocated)
Localization:   English (United States)
Keyboard layout:        English (United States)
```

Figure 11
HoneyPot VM

For the purposes of the project, Dionaea was used as a honeypot focused on web ports and DionaeaFR, which is the tool that allows me to see Dionaea graphically. After several days of our honeypot did not manage to capture anything by itself. Then I gave myself the task of analyzing what the problem was.

After reading a lot I can see that in order to obtain a comparable amount of data or enough to analyze on a large scale it takes a long time for the computer to be infected.

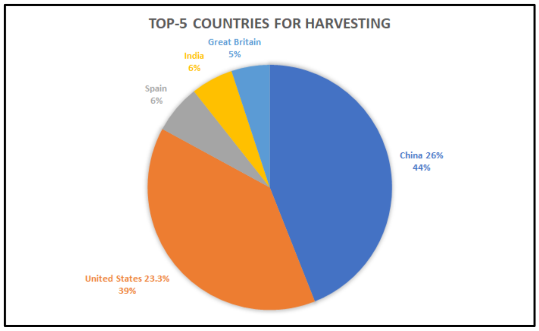


Figure 12
Project HoneyPot (COUNTRIES)

For example, if we check one of the most famous Honeypot projects "Project HoneyPot" we can see that it has been online for more than 3 years.

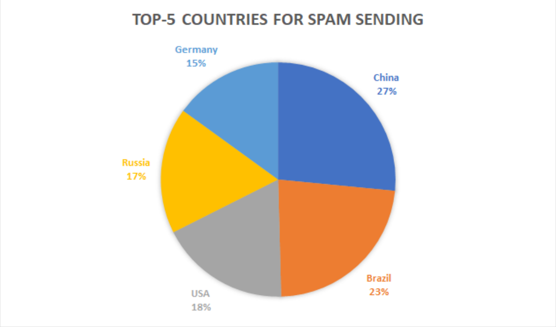
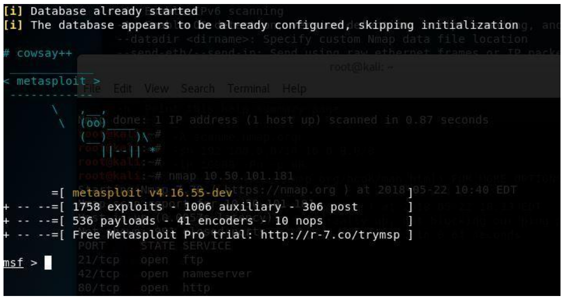


Figure 13
Project HoneyPot (SPAM)

But this was not a problem because as, I mentioned earlier I had this already planned and this is where the tools of KALI LINUX come in to verify how my honeypot behaved before an attack.

Attacker Setup

For this stage we only need to configure the device that will perform the attacks if the honeypot fails to collect data, as we mentioned in the initial stage.



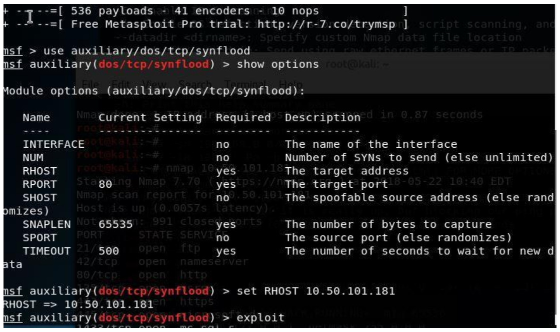
```
[i] Database already started
[i] The database appears to be already configured, skipping initialization

# cowsay++
 ____________
< metasploit >
 ------------
       \   ,__,
        \  (oo)____
           (__)    )\
              ||--|| *


       =[ metasploit v4.16.55-dev                         ]
+ -- --=[ 1758 exploits - 1006 auxiliary - 306 post       ]
+ -- --=[ 536 payloads - 41 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > 
```

Figure 14
Metasploit Installation



```
+ -- --=[ 536 payloads - 41 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use auxiliary/dos/tcp/synflood
msf auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOST                       yes       The target address
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else rand
omizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new d
ata

msf auxiliary(dos/tcp/synflood) > set RHOST 10.50.101.181
RHOST => 10.50.101.181
msf auxiliary(dos/tcp/synflood) > exploit
```

Figure 15
Metasploit DDOS

For the tests, the tools (nmap and metasploit) will be used.

Results

For this stage of results, as we had foreseen, our HoneypPot did not manage to collect the necessary data for our project and it is time to implement stage D and attack our HoneyPot and see its behavior.

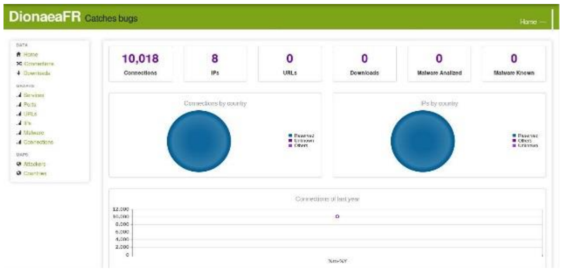


Figure 16
HoneyPot Connection Results

After completing the attack on our vulnerable machine we were able to obtain 10,018 events reported by our HoneyPot with the highest concentration of events on port 8080. What was expected because it is a vulnerable HoneyPot in this type of ports contrary to distributions like KIPPO that are focused on ssh port (22).

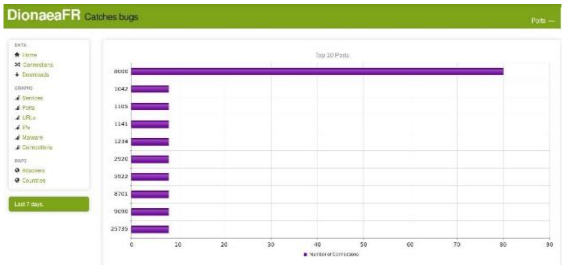
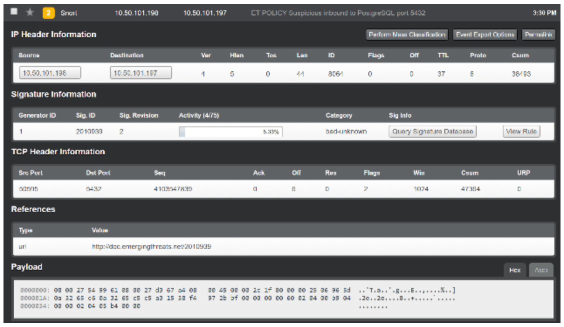


Figure 17
HoneyPot Top Protocol (8080)

Figure 18
HoneyPot Events Logs

Another advantage of using the HoneyPot with its graphical interface is that it provides us with a list of the events as if it were a type of IDS but we have to take into account that the signatures are not the same so they will never have a continuity of events. We have to understand that they are different but useful tools in a particular approach.

On the other hand we have the events registered by the SNORT in which we can obtain more information about the particular events thanks to the large number of signatures that exist in the community.



Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
2	Snort	10.50.101.198	10.50.101.197	ET POLICY Suspicious inbound to PostgreSQL port 5432	3:30 PM

IP Header Information

Source	Destination	Ver	Hlen	Tos	Len	ID	Flags	Off	TTL	Proto	Csum
10.50.101.198	10.50.101.197	4	5	0	44	8064	0	0	37	6	38493

Signature Information

Generator ID	Sig. ID	Sig. Revision	Activity (4/75)	Category	Sig Info
1	2010939	2	5.33%	bad-unknown	

TCP Header Information

Src Port	Dst Port	Seq	Ack	Off	Res	Flags	Win	Csum	URP
50505	5432	4103647839	0	6	0	2	1024	47364	0

References

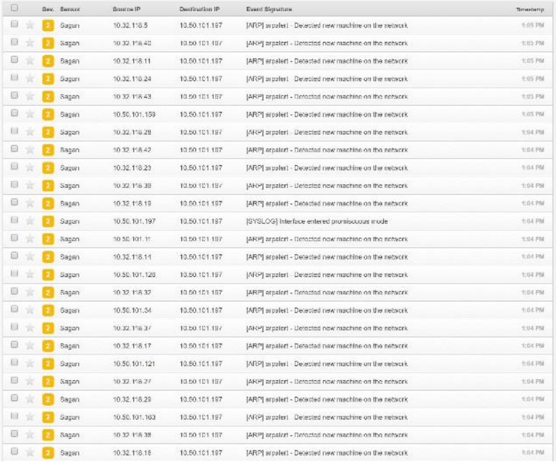
Type	Value
url	http://doc.emergingthreats.net/2010939

Payload

```
0000000: 08 00 27 54 59 61 08 00 27 d3 67 a4 08   00 45 00 00 2c 1f 80 00 00 25 06 96 5d  ..'T.a..'.g...E..,....%..]
000001A: 0a 32 65 c6 0a 32 65 c5 c5 a3 15 38 f4   97 2b bf 00 00 00 00 60 02 04 00 b9 04  .2e..2e....8..+.....`.....
0000034: 00 00 02 04 05 b4 00 00                                                           ........
```

Figure 19
Snort Payload

Thanks to the implementation of SNORBY, events can be visualized in an organized manner, which allows us to create analysis patterns and graphs. Another important point that allows us to use the SAGAN sensor together with the SNORT sensor to analyze the data. It is important to know that there are thousands of correlation and analysis tools.



Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
2	Sagan	10.32.118.5	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.32.118.40	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.32.118.11	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.32.118.24	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.32.118.43	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.50.101.159	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:05 PM
2	Sagan	10.32.118.29	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.42	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.23	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.39	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.19	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.50.101.197	10.50.101.197	[SYSLOG] Interface entered promiscuous mode	1:04 PM
2	Sagan	10.50.101.11	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.14	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.50.101.128	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.32	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.50.101.34	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.37	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.17	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.50.101.121	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.27	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.28	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.50.101.163	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.38	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM
2	Sagan	10.32.118.18	10.50.101.197	[ARP] arpalert - Detected new machine on the network	1:04 PM

Figure 20
Snort Events Logs

As part of the results obtained. We will start with the comparison of the sensors obtained in the relation of events vs time. What were very similar during a time constant but at the time of a real time DDOS using Metasploit we can see how the sagan sensor responds with a greater number of events which also includes false positives.

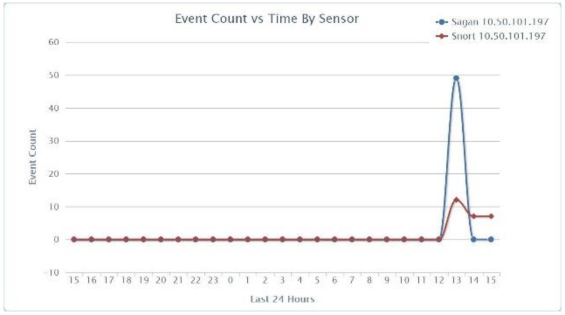


Figure 21
Events vs Time Graph

As part of the analysis, we made a comparative table of severity in order to observe the severity of the registered behavior. As expected, the sensors maintained a constant low / high level of severity until the completion of the DDOS where a peak of medium severity was recorded.

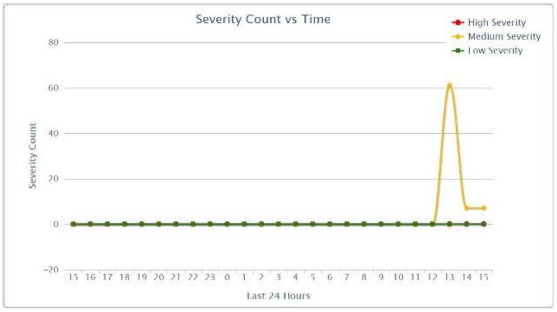


Figure 22
Severity vs Time Graph

Another important analysis was the verification of the attack sensors. In this way we were able to reach the conclusion that the signature with the most record was [ARP] arpalert with 61% over the others.

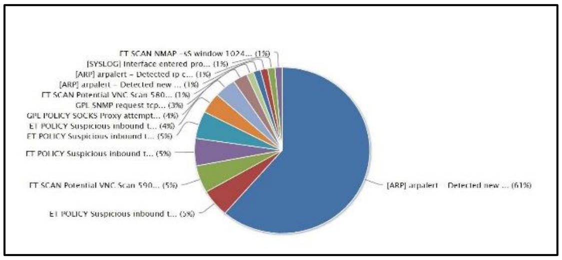


Figure 23
Top IDS Signatures Graph

No less important, it was the registry of IP addresses that attacked the system. In this case it was already expected that the IP (10.50.101.198) was in the number 1 position. This is due to the attacks made with this IP from Kali Linux.

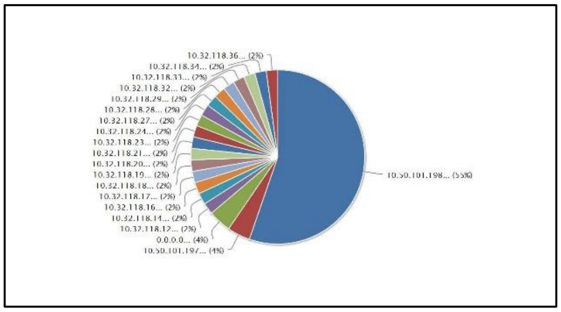
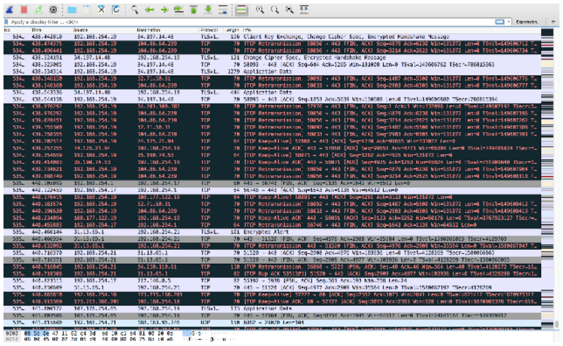


Figure 24
Attackers IP Graph

One of the easiest way to deceive a user is by giving a free Wi-Fi signal just like the food and demo's places do. Users think that they are safe but do not know how easy they can be deceived without knowing it. For this reason the configured ports mirror interface is powered by an access point to which I will remotely connect pretending to be a regular user.

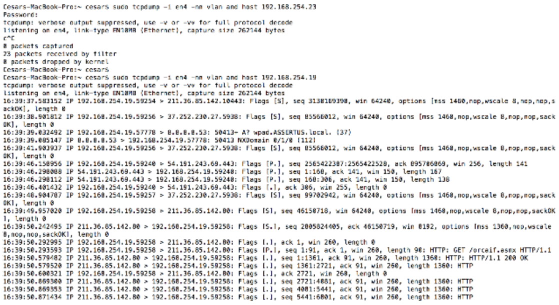


```
No.   Time        Source           Destination      Protocol Length Info
534…  438.442810  192.168.254.19   34.197.14.48     TLSv1…   106 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
534…  438.474375  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58058 → 443 [FIN, ACK] Seq=4879 Ack=6230 Win=131072 Len=0 TSval=149606712 T…
534…  438.496441  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58056 → 443 [FIN, ACK] Seq=3214 Ack=2625 Win=131072 Len=0 TSval=149606734 T…
534…  438.524191  34.197.14.48     192.168.254.19   TLSv1…   121 Change Cipher Spec, Encrypted Handshake Message
534…  438.525005  192.168.254.19   34.197.14.48     TCP       78 58093 → 443 [ACK] Seq=644 Ack=5265 Win=131008 Len=0 TSval=149606762 TSecr=786815363
534…  438.534314  192.168.254.19   34.197.14.48     TLSv1…  1279 Application Data
534…  438.546150  192.168.254.19   52.71.50.31      TCP       78 [TCP Retransmission] 58092 → 443 [FIN, ACK] Seq=1487 Ack=5500 Win=131072 Len=0 TSval=149606776 T…
534…  438.546160  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58033 → 443 [FIN, ACK] Seq=2983 Ack=6505 Win=131072 Len=0 TSval=149606777 T…
534…  438.643336  34.197.14.48     192.168.254.19   TLSv1…   444 Application Data
534…  438.644156  192.168.254.19   34.197.14.48     TCP       78 58093 → 443 [ACK] Seq=1853 Ack=5639 Win=130880 Len=0 TSval=149606880 TSecr=786815394
534…  438.970292  192.168.254.19   54.201.180.102   TCP       78 [TCP Retransmission] 57970 → 443 [FIN, ACK] Seq=1 Ack=1 Win=132096 Len=0 TSval=149607192 TSecr=1…
534…  438.970292  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58058 → 443 [FIN, ACK] Seq=4879 Ack=6230 Win=131072 Len=0 TSval=149607196 T…
534…  439.020451  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58056 → 443 [FIN, ACK] Seq=3214 Ack=2625 Win=131072 Len=0 TSval=149607262 T…
534…  439.155380  192.168.254.19   52.71.50.31      TCP       78 [TCP Retransmission] 58092 → 443 [FIN, ACK] Seq=1487 Ack=5500 Win=131072 Len=0 TSval=149607388 T…
534…  439.158355  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58033 → 443 [FIN, ACK] Seq=2983 Ack=6505 Win=131072 Len=0 TSval=149607391 T…
534…  439.207517  192.168.254.19   74.125.21.94     TCP       64 [TCP Keep-Alive] 57868 → 443 [ACK] Seq=4770 Ack=99655 Win=131072 Len=0
534…  439.257255  74.125.21.94     192.168.254.19   TCP       78 [TCP Keep-Alive ACK] 443 → 57868 [ACK] Seq=99655 Ack=4771 Win=65304 Len=0 TSval=749401024 TSecr=…
534…  439.354659  192.168.254.19   35.190.74.53     TCP       64 [TCP Keep-Alive] 58071 → 443 [ACK] Seq=1262 Ack=4925 Win=131072 Len=0
534…  439.414885  35.190.74.53     192.168.254.19   TCP       78 [TCP Keep-Alive ACK] 443 → 58071 [ACK] Seq=4925 Ack=1263 Win=65024 Len=0 TSval=751906048 TSecr=1…
535…  439.734621  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58058 → 443 [FIN, ACK] Seq=4879 Ack=6230 Win=131072 Len=0 TSval=149607964 T…
535…  439.888749  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58056 → 443 [FIN, ACK] Seq=3214 Ack=2625 Win=131072 Len=0 TSval=149608116 T…
535…  440.106009  192.168.254.1    192.168.254.17   TCP       60 443 → 56746 [FIN, ACK] Seq=1135 Ack=1643 Win=6502 Len=0
535…  440.122459  192.168.254.17   192.168.254.1    TCP       64 56746 → 443 [ACK] Seq=1643 Ack=1136 Win=64512 Len=0
535…  440.176415  192.168.254.19   108.177.122.19   TCP       64 [TCP Keep-Alive] 58091 → 443 [ACK] Seq=3261 Ack=3123 Win=131072 Len=0
535…  440.183574  192.168.254.19   52.71.50.31      TCP       78 [TCP Retransmission] 58092 → 443 [FIN, ACK] Seq=1487 Ack=5500 Win=131072 Len=0 TSval=149608412 T…
535…  440.190630  192.168.254.19   104.86.64.239    TCP       78 [TCP Retransmission] 58033 → 443 [FIN, ACK] Seq=2983 Ack=6505 Win=131072 Len=0 TSval=149608419 T…
535…  440.234094  108.177.122.19   192.168.254.19   TCP       78 [TCP Keep-Alive ACK] 443 → 58091 [ACK] Seq=3123 Ack=3262 Win=58176 Len=0 TSval=3767923127 TSecr=…
535…  440.455883  192.168.254.17   192.168.254.1    TCP       64 [TCP Retransmission] 56746 → 443 [FIN, ACK] Seq=1643 Ack=1136 Win=64512 Len=0
535…  440.460144  31.13.65.1       192.168.254.21   TLSv1…   101 Encrypted Alert
535…  440.466954  31.13.65.1       192.168.254.21   TCP       78 443 → 51520 [FIN, ACK] Seq=4976 Ack=2908 Win=35584 Len=0 TSval=1580066865 TSecr=4128269
535…  440.632092  31.13.65.1       192.168.254.21   TCP       78 [TCP Retransmission] 443 → 51520 [FIN, ACK] Seq=4976 Ack=2908 Win=35584 Len=0 TSval=1580067047 T…
535…  440.716370  192.168.254.21   31.13.65.1       TCP       78 51520 → 443 [ACK] Seq=2908 Ack=4976 Win=103936 Len=0 TSval=4128269 TSecr=1580066865
535…  440.716371  192.168.254.21   31.13.65.1       TCP       78 51520 → 443 [FIN, ACK] Seq=2908 Ack=4977 Win=103936 Len=0 TSval=4128269 TSecr=1580066865
535…  440.716845  192.168.254.21   34.230.110.61    TCP      118 [TCP Retransmission] 50808 → 5223 [PSH, ACK] Seq=48 Ack=46 Win=364 Len=48 TSval=4128272 TSecr=16…
535…  440.718306  192.168.254.21   31.13.65.1       TCP       82 [TCP Dup ACK 53513#1] 51520 → 443 [ACK] Seq=2909 Ack=4977 Win=103936 Len=0 TSval=4128288 TSecr=1…
535…  440.823313  192.168.254.17   217.146.0.5      TCP       82 55192 → 5938 [PSH, ACK] Seq=361 Ack=193 Win=258 Len=24
535…  440.838089  31.13.65.1       192.168.254.21   TCP       78 443 → 51520 [ACK] Seq=4977 Ack=2909 Win=35584 Len=0 TSval=1580067197 TSecr=4128289
535…  440.865010  192.168.254.18   173.215.186.201  TCP       78 [TCP Keep-Alive] 57727 → 80 [ACK] Seq=2352 Ack=3073 Win=2641 Len=0 TSval=821412 TSecr=1678975312
535…  440.915369  173.215.186.201  192.168.254.18   TCP       78 [TCP Keep-Alive ACK] 80 → 57727 [ACK] Seq=3073 Ack=2353 Win=128 Len=0 TSval=1678981376 TSecr=816…
535…  441.008172  107.178.254.65   192.168.254.19   TLSv1…   133 Application Data
535…  441.003586  107.178.254.65   192.168.254.19   TCP       78 443 → 57964 [FIN, ACK] Seq=8754 Ack=1945 Win=64512 Len=0 TSval=24041184 TSecr=149608917
535…  441.015680  192.168.254.21   108.161.95.246   UDP      150 5002 → 25020 Len=104

0000  08 5d 0e 47 11 62 c4 9d  ed 10 c1 e4 81 02 20 0a
0010  00 00 45 00 02 2d 05 c9  40 00 00 06 75 0d c0 a8
```

Figure 25
WireShark Capture

In just 4 minutes you can get a capture of over 53,000 packages. Using the Wireshark tool. Captures were also made at the terminal level with the TCPDUMP command **(sudo tcpdump -i eh4 -nnvvv vlan and host 192.168.254.23**). These captures were in PCAP format which allows us to analyze them with the different data analysis programs with the same Wireshark. It is important to mention that the interface used was l eht4 which is configured as promiscuous for monitoring.



```
Cesars-MacBook-Pro:~ cesar$ sudo tcpdump -i en4 -nn vlan and host 192.168.254.23
Password:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en4, link-type EN10MB (Ethernet), capture size 262144 bytes
c^C
0 packets captured
23 packets received by filter
0 packets dropped by kernel
Cesars-MacBook-Pro:~ cesar$
Cesars-MacBook-Pro:~ cesar$ sudo tcpdump -i en4 -nn vlan and host 192.168.254.19
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on en4, link-type EN10MB (Ethernet), capture size 262144 bytes
16:39:37.583152 IP 192.168.254.19.59254 > 211.36.85.142.10443: Flags [S], seq 3138169398, win 64240, options [mss 1460,nop,wscale 8,nop,nop,s
ackOK], length 0
16:39:38.901812 IP 192.168.254.19.59256 > 37.252.230.27.5938: Flags [S], seq 85568012, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sack
OK], length 0
16:39:39.032492 IP 192.168.254.19.57778 > 8.8.8.8.53: 50413+ A? wpad.ASSERTUS.local. (37)
16:39:39.085147 IP 8.8.8.8.53 > 192.168.254.19.57778: 50413 NXDomain 0/1/0 (112)
16:39:41.903937 IP 192.168.254.19.59256 > 37.252.230.27.5938: Flags [S], seq 85568012, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sack
OK], length 0
16:39:46.150956 IP 192.168.254.19.59240 > 54.191.243.69.443: Flags [P.], seq 2565422387:2565422528, ack 895786869, win 256, length 141
16:39:46.290008 IP 54.191.243.69.443 > 192.168.254.19.59240: Flags [P.], seq 1:168, ack 141, win 150, length 167
16:39:46.290112 IP 54.191.243.69.443 > 192.168.254.19.59240: Flags [P.], seq 168:306, ack 141, win 150, length 138
16:39:46.401432 IP 192.168.254.19.59240 > 54.191.243.69.443: Flags [.], ack 306, win 255, length 0
16:39:48.904707 IP 192.168.254.19.59257 > 37.252.230.27.5938: Flags [S], seq 99702942, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sack
OK], length 0
16:39:49.957020 IP 192.168.254.19.59258 > 211.36.85.142.80: Flags [S], seq 46150718, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sackOK
], length 0
16:39:50.242495 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [S.], seq 2005824405, ack 46150719, win 8192, options [mss 1360,nop,wscale
8,nop,nop,sackOK], length 0
16:39:50.292995 IP 192.168.254.19.59258 > 211.36.85.142.80: Flags [.], ack 1, win 260, length 0
16:39:50.293593 IP 192.168.254.19.59258 > 211.36.85.142.80: Flags [P.], seq 1:91, ack 1, win 260, length 90: HTTP: GET /orcaif.asmx HTTP/1.1
16:39:50.579482 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [.], seq 1:1361, ack 91, win 260, length 1360: HTTP: HTTP/1.1 200 OK
16:39:50.579520 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [.], seq 1361:2721, ack 91, win 260, length 1360: HTTP
16:39:50.600321 IP 192.168.254.19.59258 > 211.36.85.142.80: Flags [.], ack 2721, win 260, length 0
16:39:50.869300 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [.], seq 2721:4081, ack 91, win 260, length 1360: HTTP
16:39:50.869353 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [.], seq 4081:5441, ack 91, win 260, length 1360: HTTP
16:39:50.871434 IP 211.36.85.142.80 > 192.168.254.19.59258: Flags [.], seq 5441:6801, ack 91, win 260, length 1360: HTTP
```

Figure 26
tcpdump Capture

SUMMARY AND CONCLUSIONS

In conclusion mode, I can conclude by reaffirming that time is the key to investigations. The initial motive of this project was the creation of the honeypot with the hope of being able to analyze natural events in which real patterns could be created on which to base research and develop new defenses. In addition, the use of DIONAEA and SNORT was phenomenal because it can interact with two powerful tools. Even with this fact, I can analyze the functionality of the tools and their response to common attacks. I cannot say that it was a failure because the amount of information that can be learned was immense what from the professional point of view made me increase my level of knowledge in areas where I had not been involved. In conclusion it was a great learning and implementation experience.

FUTURE WORK

As a future plan, I want to be able to establish the honeypot at a level of development in which I can be exposed to the internet for more time in order to obtain more accurate data on which I can develop appropriate security methods. In addition to my future plans is the use of other detection tools such as SURICATA and SPLUNK to integrate syslog services. I would also like to integrate other Honeypots with different vulnerabilities like KIPPO for SHH we're brute-force attacks are very common.

ACKNOWLEDGEMENTS

I would like to acknowledge all the people that contributed on the survey and to Dr. Jeffrey Duffany with his guidance during the project.

REFERENCES

[1] K. Apostol, *Brute-force Attack*, Salu Publishing, 2012.

[2] I. Pranata, G. Skinner & R. Athauda, "A community based authentication and authorization mechanism for digital ecosystem," in *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, Daejeon, 2011, pp. 158-163.

[3] D. J. Barrett and R. E. Silverman, *SSH, the Secure Shell: the definitive guide*, O'Reilly Media, Inc., 2001.

[4] D. McGrew and B. Anderson, "Enhanced telemetry for encrypted threat analytics," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, Singapore, 2016, pp. 1-6.