PROJECT: SHADOW SENTRY

Phua Tong Huat

Trainer: James Lim

Table of Contents

Introduction

Shadow Sentry Project

In this project, we are to create an Elastic Cloud system as an Security Operations Center(SOC) to detect, analyze and monitor malicious activities on our network. We will also be creating pentest scripts to test our SOC's monitoring and alert system.

The main objectives of the project are:

1. Create a Security Operations Center (SOC) system by installing Elastic Cloud and Honeypot on Digital Ocean to detect and analyze malicious activities on our network
2. Develop pentest scripts to attack the honeypot
3. Monitor the alerts

SOC is crucial in a organization's defense against cyber threats in real time as a robust SOC will be able to detect, analyze and respond to security incidents and minimize damages/losses to the organization. Therefore it is important that the monitoring system that we set up is able to maintain vigilance over the organization's networks, systems and applications at all time.

Methodologies

To set up and test the SOC system, the following was done:

1. Set up Elastic Cloud on a server using DigitalOcean(DO)
2. Set up Honeypot Cowrie on another server using DO
3. Link the two servers and ensured that there are proper communications
4. Create attack scripts to test the honeypot and also to see the events being captured in ELK
5. Alerting and Monitoring

Methodologies

1. **Setting up Elastic Cloud on a server using DigitalOcean [1]**

   A server named SOC was created on DO and the ELK stack, version 7.17.18 was installed on it. Full details can be retrieved from the installation guide attached.

   ELK – Elasticsearch, Logstash, Kibana is a set of technologies to collect, search, analyze and visualize data collected from various source. The system was set up in such a way that data from designated source will be sent over from Beats to Logstash and in turn, Logstash will send the data to Elasticsearch. Kibana will be used to visualize the data using visual charts such as graphs, histograms and pie charts etc.

2. **Set up Honeypot Cowrie on another server using DigitalOcean [2]**

   A 2nd server (HP) was created on DO and Honeypot Cowrie is installed. Cowrie is a medium interaction SSH and Telnet honeypot designed to log brute force attacks and also capture shell commands executed by the attacker. HP server was harden up by implementing strong login credentials and firewall was also configured to restrict access by known IP address only. Only essential ports were opened as well.

   Filebeat was installed on HP server to collect the logs and send to Logstash. Auditd module was also installed to enable the capturing of audit logs.

```
root@ubuntu-hp:/home/cowrie/cowrie/var/log/cowrie# ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
23/tcp                     ALLOW       Anywhere
22222/tcp                  ALLOW       Anywhere
2222/tcp                   ALLOW       Anywhere
2223/tcp                   ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
443/tcp                    DENY        Anywhere
80                         DENY        Anywhere
Anywhere                   ALLOW       118.200.202.190
OpenSSH (v6)               ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
23/tcp (v6)                ALLOW       Anywhere (v6)
22222/tcp (v6)             ALLOW       Anywhere (v6)
2222/tcp (v6)              ALLOW       Anywhere (v6)
2223/tcp (v6)              ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
443/tcp (v6)               DENY        Anywhere (v6)
80 (v6)                    DENY        Anywhere (v6)
```

Fig 1. UFW status on HP server showing restricted access

Methodologies

3. Link the two servers and ensured that there are proper communications between them

```
# ============================== Filebeat inputs ===============================

filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.

# filestream is an input for collecting log messages from files.
- type: filestream

  # Unique ID among all inputs, an ID is required.
  id: my-filestream-id

  # Change to true to enable this input configuration.
  enabled: false

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/*.log
#    - /var/log/vsftpd/vsftpd.log
- type: log
  enabled: true
  paths:
    - /home/cowrie/cowrie/var/log/cowrie/cowrie.json*
  fields:
    event.type: cowrie

#- type: log
#  paths:
#    - /var/log/vsftpd/vsftpd.log*
#- c:\programdata\elasticsearch\logs\*
```

Fig 2 Filebeat.yml from HP server showing the config added to collect logs from Cowrie

Fig 2 shows the file beat config file, filebeat.yml. This will instruct filebeat to extract the log in /var/log and also the cowrie logs and send over to logstash.

Methodologies

3. Link the two servers and ensured that there are proper communications between them

```
GNU nano 4.8                                    logstash-cowrie.conf

input {
      # filebeats
      beats {
            port => 5044
            type => "cowrie"
      }
}

filter {
   if [type] == "cowrie" {
       json {
           source => message
           target => honeypot
       }

       date {
           match => [ "timestamp", "ISO8601" ]
       }

       if [src_ip]  {

           mutate {
               add_field => { "src_host" => "%{src_ip}" }
           }

           dns {
               reverse => [ "src_host" ]
               nameserver => [ "8.8.8.8", "8.8.4.4" ]
               action => "replace"
               hit_cache_size => 4096
               hit_cache_ttl => 900
               failed_cache_size => 512
               failed_cache_ttl => 900
           }


           geoip {
               source => "src_ip"
               target => "geoip"
               database => "/usr/share/logstash/vendor/geoip/GeoLite2-City.mmdb"
               add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}"]
               add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ]
           }
```

Fig 3.1 logstash-cowrie.conf

Fig 3.1 and Fig 3.2 shows the logstash-cowrie.conf[3] file in logstash. The config file will instruct Logstash how to parse the logs and output according to our requirement

Methodologies

3. Link the two servers and ensured that there are proper communications between them

```
  GNU nano 4.8                                                    logstash-cowrie.conf
                hit_cache_ttl => 900
                failed_cache_size => 512
                failed_cache_ttl => 900
            }


            geoip {
                source => "src_ip"
                target => "geoip"
                database => "/usr/share/logstash/vendor/geoip/GeoLite2-City.mmdb"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}"]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ]
            }

        }

        mutate {
            # cut out useless tags/fields
            remove_tag => [ "beats_input_codec_plain_applied"]
            remove_field => [ "[log][file][path]", "[log][offset]" ]
            convert => [ "[geoip][coordinates]", "float" ]
        }
    }
}

output {
    if [type] == "cowrie" {
        elasticsearch {
            hosts => ["localhost:9200"]
            cacert => '/etc/logstash/certs/elastic-certificates.p12'
            user => "elastic"
            password => "password"
            ilm_enabled => auto
            ilm_rollover_alias => "cowrie-logstash"
        }
        file {
            path => "/tmp/cowrie-logstash.log"
            codec => json
        }
        stdout {
            codec => rubydebug
        }
    }
}
```

Fig 3.2 logstash-cowrie.conf

Fig 3.1 and Fig 3.2 shows the logstash-cowrie.conf[3] file in logstash. The config file will instruct Logstash how to parse the logs and output according to our requirement

Methodologies

3. Link the two servers and ensured that there are proper communications between them

```
  GNU nano 4.8                                  30-elasticsearch-output.conf
output {
  if [@metadata][pipeline] {
        elasticsearch {
        hosts => ["localhost:9200"]
        cacert => '/etc/logstash/certs/elastic-certificates.p12'
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        pipeline => "%{[@metadata][pipeline]}"
        user => "elastic"
        password => "password"
        }
  } else {
        elasticsearch {
        hosts => ["localhost:9200"]
        cacert => '/etc/logstash/certs/elastic-certificates.p12'
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        user => "elastic"
        password => "password"
        }
  }
}
```

Fig 4. 30-elasticseach-output.conf

Fig 4 shows the elasticsearch config file in Logstash. Point to take note is the addition of the fields “cacert”, “user” and “password” to ensure that the 2 servers can communicate.

Methodologies

3. Link the two servers and ensured that there are proper communications between them

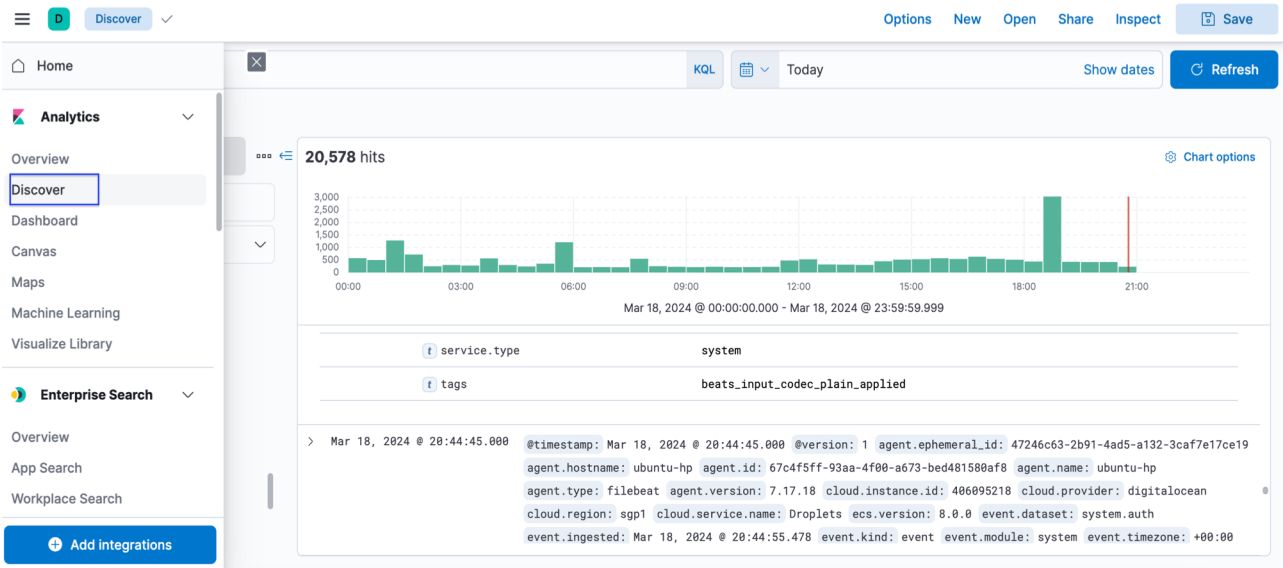

Fig 5. Kibana interface

If the connection between servers SOC and HP is successful, a similar interface such as fig 5 will appear when connected to http://localhost:5601.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

An attack script with 3 different attacks were created to test the HP server. The 3 attacks are:

a. Brute Force Attack
Brute force attack is to use trial and error to crack passwords and login credentials to gain access to port 22(SSH) and 23(Telnet) of the intended server. SSH and Telnet are 2 of the most commonly used ports and there is a high probability that these ports are left opened by most users. If the user had a weak login credentials, we would be able to obtain the credentials using brute force and carry out further exploitation. In this attack, the script will do a brute force attack on port 22 and 23 using Medusa tool. For the purpose of this project, a short default username and password list were used to reduce the waiting time for the brute force to be completed.

b. DDOS Attack [4]
Distributed denial-of-service (DDOS) attack is to disrupt the normal traffic of a targeted server, service or network by overwhelming the target with a flood of Internet traffic. In this attack, the script will first check for hping3 and use if the module is available in the user's system. If not, nping would be used instead. User will also be asked for inputs like whether to send data packets to destination IP and which source IP to use.

c. FTP Anon Attack [5]
FTP Anon Attack is to attempt login to FTP server using anonymous user account and download all the files from the ftp server to see if any valuable information is left on the unsecured directory. In many cases, some servers allow FTP anonymous login for easier work flow and file sharing but this is actually very risky as bad actors will have a chance to exploit the servers. Some users also might upload sensitive data onto the FTP servers and these data will become at risk of being exposed.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK
-Ethical use warnings and system checks

```
└─$ bash socProjectUpdated.sh
WARNING: This script is for educational and testing purposes only.
It is illegal to use this script to attack or penetrate systems without
explicit permission from the system owner. You are responsible for
complying with all applicable laws and regulations.

By running this script, you acknowledge that you have obtained proper
authorization and are using it for legitimate purposes only.
Enter 1 to proceed with the script. Any other inputs will exit this script.
1
Proceeding with the script now..
Nmap is installed.

Please enter network or IP address to attack:
```

Fig 6. Ethical warning and system checks before start of script

```
# function to check that Nmap is installed
function checkNmapStatus()
{
# check if nmap is installed, proceed if installed. prompt user to install nmap and exit if not installed
if command -v nmap &> /dev/null;
then
    echo "Nmap is installed."
    getNetwork
else
    echo "Nmap is not installed. You can install it by running 'sudo apt-get install nmap'."
    exit 1
fi
}
```

As seen in Fig 6.0, the user is informed about the usage of the script and being asked to proceed further only if user agrees to use the script for educational and testing purposes only. Ethical warning is important as we do not want user to misuse the script and perform any malicious actions with it.

The machine is also being checked if Nmap is installed as it is a necessary for the script to proceed further. If Nmap is not installed, instruction was given to install and try again.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

a. Brute Force Attack

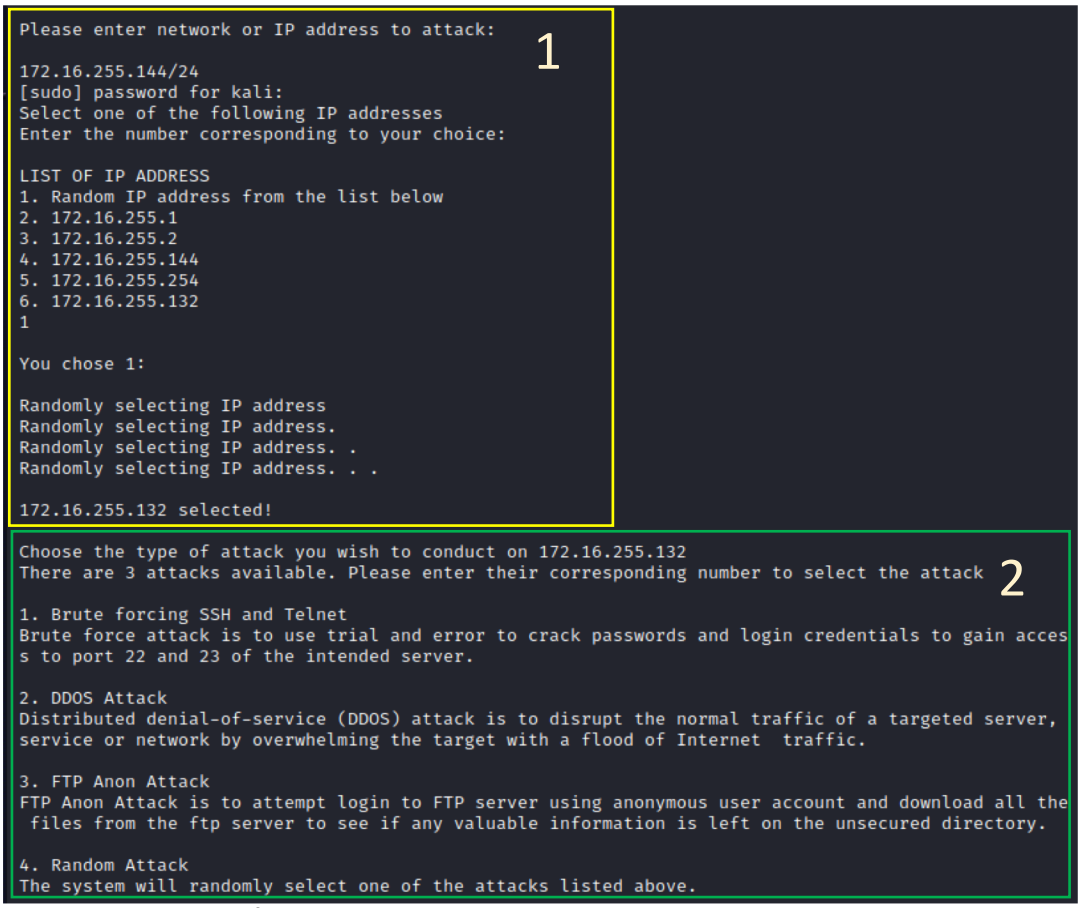

Fig 6.1 Requesting IP from user and randomly selecting IP

As seen in box 1 in Fig 6.1, user is being asked to enter either a network or an IP address to attack. Subsequently, the network will be scanned and available IP addresses will be listed as shown. User can either select any of the IP to attack or let the system randomly select 1 IP from the list.

BOX 2

The 3 attacks are listed out and again, the user has the option of choosing the attack or let the system randomly choose.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

a. Brute Force Attack

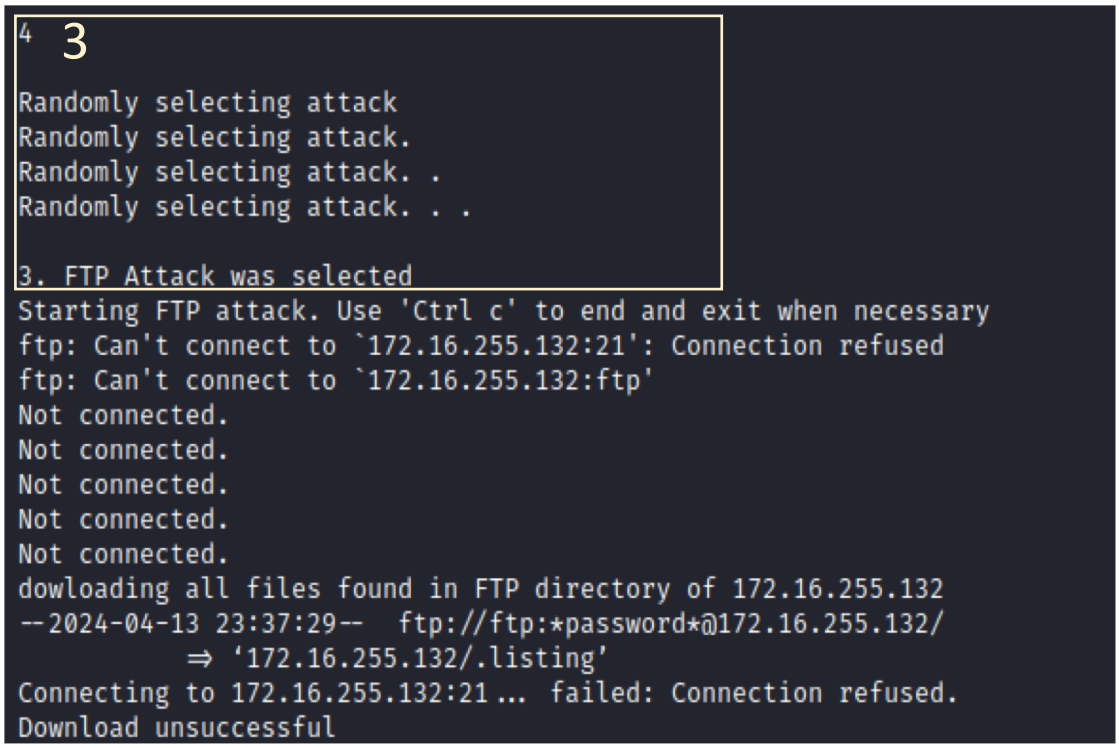

Fig 6.2 Randomly select attack type

BOX 3

User chose to randomly select an attack and thus the system will proceed to initiate an attack on the selected IP address. In the above example, the connection to the target's port 21 failed because the port was closed. Other reasons that might cause the connection to fail could be firewall rules or the machine does not allow anonymous login.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

a. Brute Force Attack

```
4. Random Attack
The system will randomly select one of the attacks listed above.

1

1. Brute Force Attack was selected

Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: root (1 of 3, 0 complete) Passw
ord:  (1 of 8 complete)
ACCOUNT FOUND: [ssh] Host: 159.223.49.30 User: root Password:  [SUCCESS]
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word:  (1 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: admin (2 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: whatever (3 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: windows (4 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: 123456 (5 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: writer (6 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: zxcvbnm (7 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: admin (2 of 3, 1 complete) Pass
word: zxczxc (8 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord:  (1 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: test (2 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: whatever (3 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: windows (4 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: 123456 (5 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: writer (6 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: zxcvbnm (7 of 8 complete)
ACCOUNT CHECK: [ssh] Host: 159.223.49.30 (1 of 1, 0 complete) User: test (3 of 3, 2 complete) Passw
ord: zxczxc (8 of 8 complete)
    **Brute force on SSH done**
    **Brute force result saved in .//SSH_bruteforce.txt
```

Fig 7.1 Brute Force on Port 22

```
ERROR: telnet.mod: Telnet did not respond to the sending of the user name 'root' in a timely fashio
n - is it down or refusing connections?
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [telnet] Host: 159.223.49.30 (1 of 1, 0 complete) User: root (1 of 3, 0 complete) Pa
ssword:  (1 of 8 complete)
    **Brute force on Telnet done
    **BRute force result saved in .//telnet_bruteforce.txt
```

Fig 7.2 Brute Force on Port 23 (unsuccessful)

Fig 7.1 shows a successful brute force attack and the result was displayed and also saved to a local directory as shown

Fig 7.2 shows a failed connection to Port 23. Failed attempts can be due to many reasons such as closed port or restricted access by firewall.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

a. Brute Force Attack

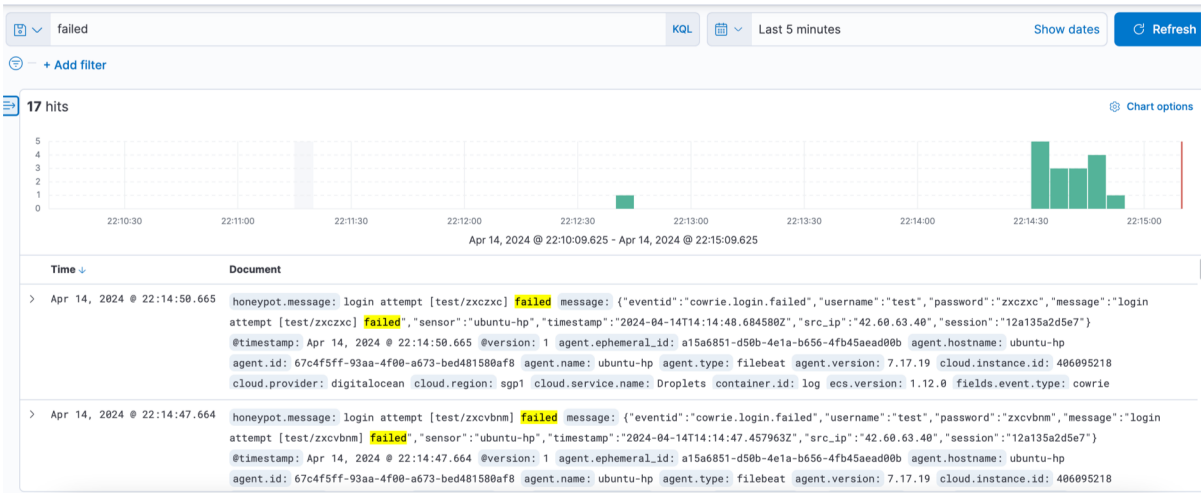

Fig 7.3 Failed SSH login attempts

Fig 7.3 shows the failed login attempts captured by the system. Details such as login user name and password used are reflected in the logs as well. This allows us as to analyze if the network is being compromised. If there are repeated attempts by the same user using different passwords to login, there is a high possibility that someone is trying to bruteforce into the network.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

b. DDOS Attack

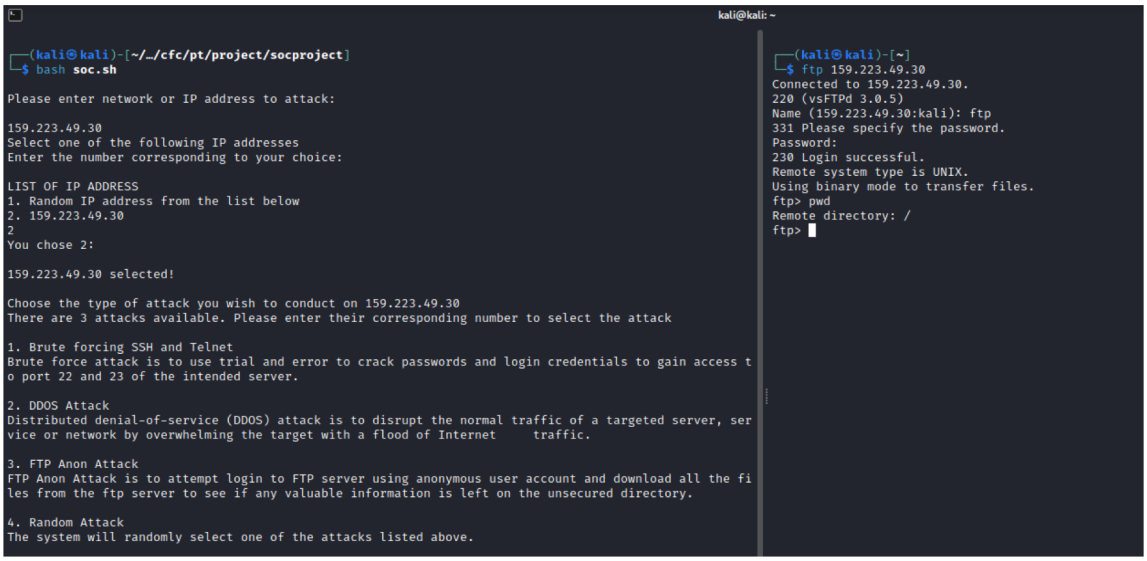

Fig 8.1 Before DDOS attack

Fig 8 shows the current connection before DDOS attack starts. The right terminal shows an active FTP connection between the 2 servers.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

b. DDOS Attack

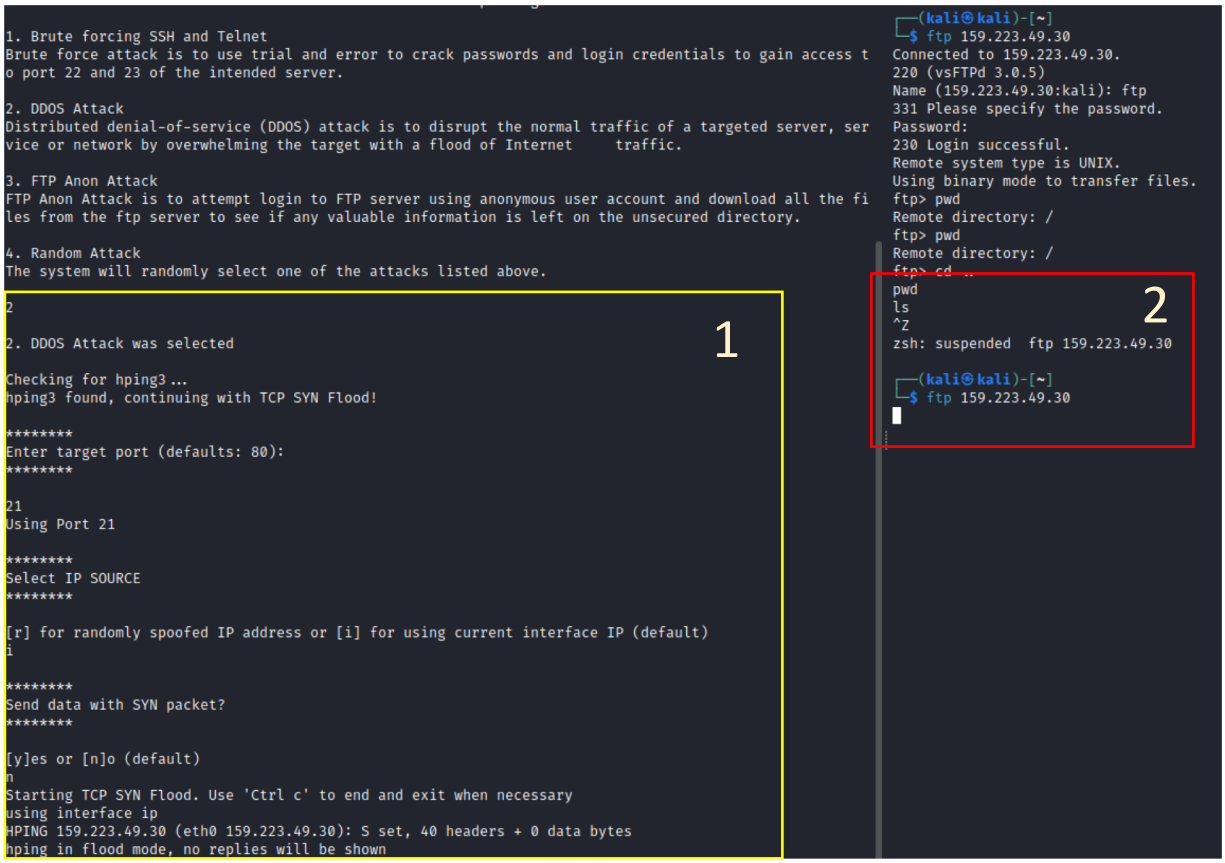

Fig 8.2 DDOS started

BOX 1

User is being asked to select the configuration for the DDOS attack. User can choose the port to attack, how much data to flood the destination IP etc. The attack will initiate once the config is done and will not stop unless cancelled by user.

BOX 2

We can see that the connection to FTP server is not working as there are no responses to the command keyed in. Subsequent attempt to establish connection failed as well. This is because the target on the right is being flooded with packets and are unable to function as intended. Figure 8.3 will show more details about the DDOS attack.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

b. DDOS Attack

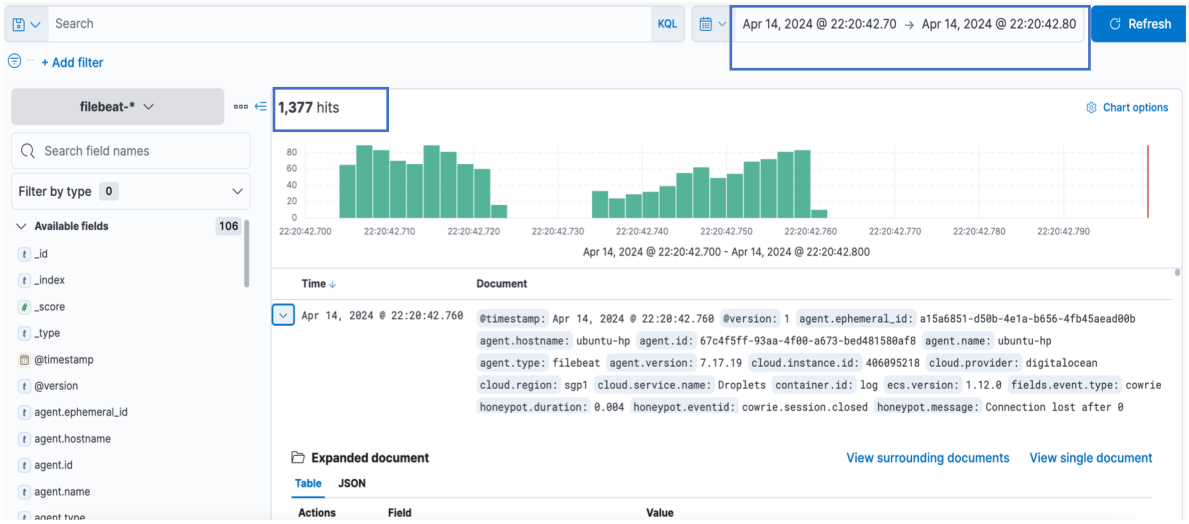

Fig 8.3 The filebeat log once DDOS started

Fig 8.3 shows ELK capturing the cowrie server being flooded with packets once DDOS starts as seen in Fig 8.2. The target received more than 1000packets in just a few seconds. This shows that DDOS is a very powerful tool to deny any machine the ability to function and if not protected properly, our network would be affected once we are being targeted.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

c. FTP Anon Attack (Unsuccessful attempt)

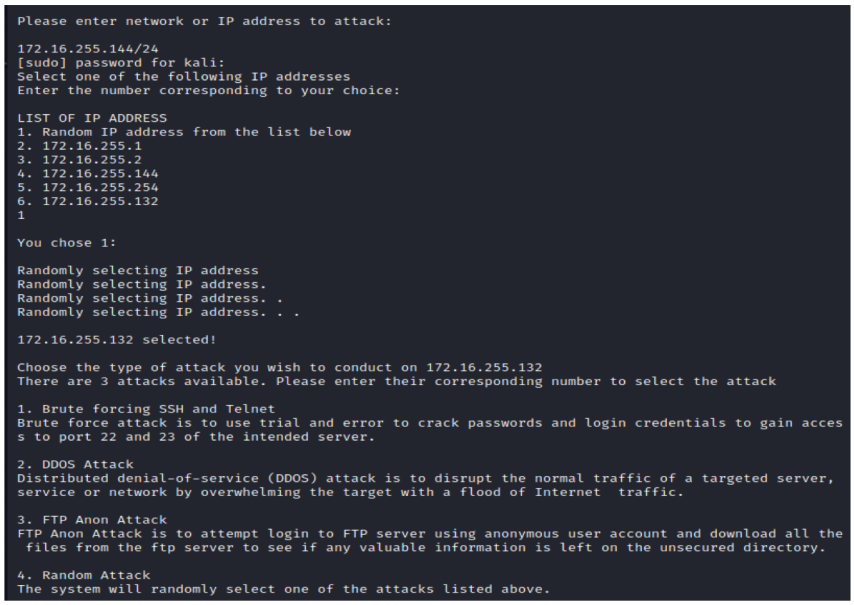

Fig 9.1 Initialization of attack where the IP address was randomly selected

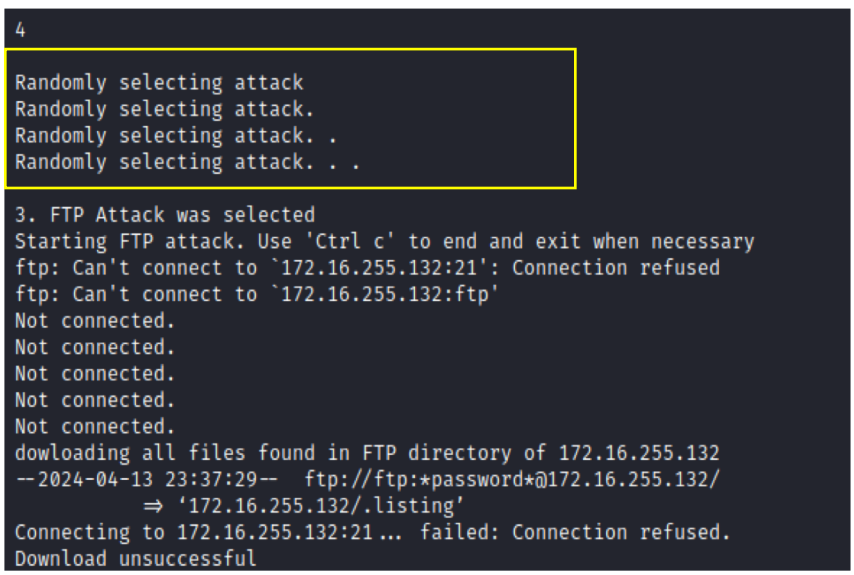

Fig 9.2 Random selection of attack and the resulting unsuccessful FTP attack

Fig 9.2 shows the outcome of an unsuccessful attack on a FTP server. Failed attempts can be due to many reasons such as closed port, anonymous login not allowed or restricted access by firewall.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

c. FTP Anon Attack

```
4. Random Attack
The system will randomly select one of the attacks listed above.

3
3. FTP Attack was selected
Starting FTP attack. Use 'Ctrl c' to end and exit when necessary
Connected to 159.223.49.30.
220 (vsFTPd 3.0.5)
331 Please specify the password.
230 Login successful.
229 Entering Extended Passive Mode (|||24414|)
150 Here comes the directory listing.
drwxrwxrwx    4 0        0            4096 Apr 13 15:55 home
226 Directory send OK.
Remote directory: /
250 Directory successfully changed.
221 Goodbye.
dowloading all files found in FTP directory of 159.223.49.30
--2024-04-13 23:55:59--  ftp://ftp:*password*@159.223.49.30/
           ⇒ '159.223.49.30/.listing'
Connecting to 159.223.49.30:21... connected.
Logging in as ftp ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD not needed.
==> PASV ... done.    ==> LIST ... done.

159.223.49.30/.listing       [ <=>                                ]     181  --.-KB/s    in 0s

2024-04-13 23:56:02 (3.16 MB/s) - '159.223.49.30/.listing' saved [181]

--2024-04-13 23:56:02--  ftp://ftp:*password*@159.223.49.30/home/
           ⇒ '159.223.49.30/home/.listing'
==> CWD (1) /home ... done.
==> PASV ... done.    ==> LIST ... done.

159.223.49.30/home/.list     [ <=>                                ]     444  --.-KB/s    in 0s

2024-04-13 23:56:04 (26.2 MB/s) - '159.223.49.30/home/.listing' saved [444]

--2024-04-13 23:56:04--  ftp://ftp:*password*@159.223.49.30/home/hello.txt
           ⇒ '159.223.49.30/home/hello.txt'
```

Fig 9.3 FTP Anonymous Attack

Fig 9.3 is executing the FTP attack by logging in to the destination IP using anonymous account ftp:ftp. This is possible because the machine allows anonymous login. Some networks allow anonymous login because they want to share files among many users easily, without any need of unique credentials to login. Allowing FTP anonymous login is very dangerous as FTP servers are continuously being scanned by automated tools and the vulnerable servers might exploited by hackers. Hackers might be able to upload malicious files onto the servers and users might download or execute these files and become compromised.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

c. FTP Anon Attack

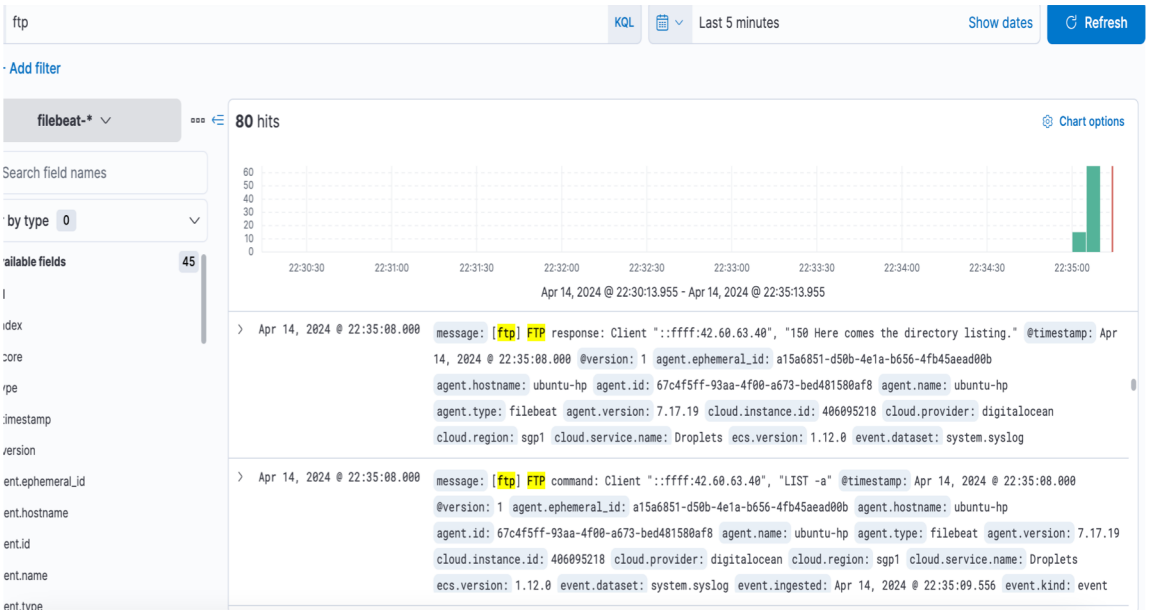

Fig 9.4 ELK showing the FTP Anon Attack

Fig 9.4 shows the actions taken by the attacker while in the FTP server. The attacker can perform read, download or upload files onto the server, depending on the settings imposed by the administrator.

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

c. FTP Anon Attack

```
--2024-04-13 23:56:21--  ftp://ftp:*password*@159.223.49.30/home/admin/.profile
           ⇒ '159.223.49.30/home/admin/.profile'
==> CWD not required.
==> PASV ... done.    ==> RETR .profile ... done.
Length: 807

159.223.49.30/home/admin 100%[=====================================>]     807  --.-KB/s    in 0s

2024-04-13 23:56:22 (11.3 MB/s) - '159.223.49.30/home/admin/.profile' saved [807]

--2024-04-13 23:56:22--  ftp://ftp:*password*@159.223.49.30/home/admin/myadminfolder/
           ⇒ '159.223.49.30/home/admin/myadminfolder/.listing'
==> CWD (1) /home/admin/myadminfolder ... done.
==> PASV ... done.    ==> LIST ... done.

159.223.49.30/home/admin     [ <=>                                ]     119  --.-KB/s    in 0s

2024-04-13 23:56:24 (7.82 MB/s) - '159.223.49.30/home/admin/myadminfolder/.listing' saved [119]

FINISHED --2024-04-13 23:56:24--
Total wall clock time: 24s
Downloaded: 14 files, 11K in 0.02s (645 KB/s)
Download successful
```

Fig 9.5 Downloading files from FTP server

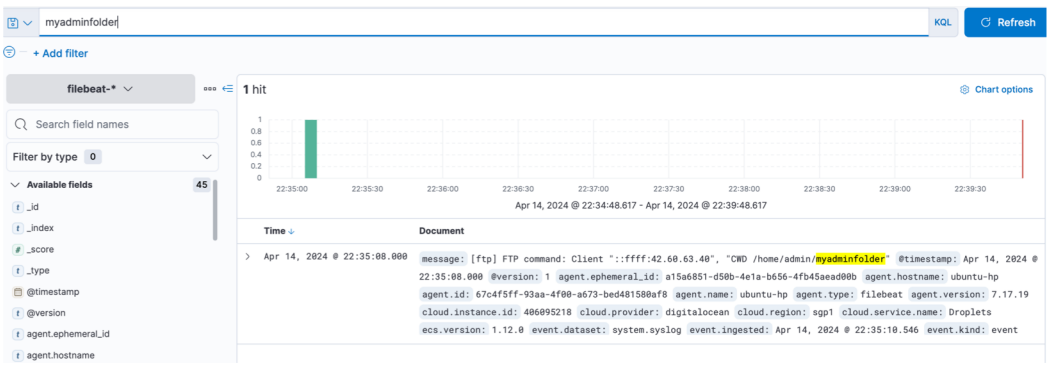

Fig 9.6 ELK capturing the moment the directory “myadminfolder” being accessed

The successful login was followed by the commencement of file download from the FTP server shown in Fig 9.5.

User can then screen through the downloaded files for any important or sensitive information that can be further exploited.

FIG 9.6 shows the action being logged by the system

Methodologies

4. Create attack scripts to test the honeypot and also to see the events being captured in ELK

d. Attack log by attacker

```
┌──(kali㉿kali)-[~/…/cfc/pt/project/socproject]
└─$ sudo cat /var/log/attack.log

09/04/2024 17:47:19, Type of attack: Brute Force on SSH and Telnet, Destination: 159.223.49.30
09/04/2024 17:49:42, Type of attack: DDOS, Destination: 159.223.49.30
11/04/2024 22:14:47, Type of attack: FTP anon login, Destination: 159.223.49.30, Result: Anonymous login successful
11/04/2024 22:41:31, Type of attack: FTP anon login, Destination: 172.16.255.132, Result: Anonymous login successful
11/04/2024 22:41:31, Type of attack: FTP directory files download, Destination: 172.16.255.132, Result: Download unsuccessful
11/04/2024 22:54:11, Type of attack: FTP anon login, Destination: 172.16.255.144
11/04/2024 23:01:58, Type of attack: FTP anon login, Destination: 172.16.255.254
11/04/2024 23:08:27, Type of attack: FTP anon login, Destination: 172.16.255.254
11/04/2024 23:23:45, Type of attack: FTP anon login, Destination: 159.223.49.30
11/04/2024 23:29:22, Type of attack: FTP anon login, Destination: 159.223.49.30
11/04/2024 23:31:53, Type of attack: Brute Force on SSH and Telnet, Destination: 159.223.49.30
11/04/2024 23:36:11, Type of attack: DDOS, Destination: 159.223.49.30
11/04/2024 23:41:05, Type of attack: DDOS, Destination: 159.223.49.30
11/04/2024 23:48:18, Type of attack: DDOS, Destination: 159.223.49.30
11/04/2024 23:52:23, Type of attack: Brute Force on SSH and Telnet, Destination: 172.16.255.144
12/04/2024 00:16:58, Type of attack: Brute Force on SSH and Telnet, Destination: 172.16.255.2
12/04/2024 00:19:06, Type of attack: Brute Force on SSH and Telnet, Destination: 172.16.255.2
13/04/2024 23:35:57, Type of attack: FTP anon login, Destination: 172.16.255.132
13/04/2024 23:35:57, Type of attack: FTP directory files download, Destination: 172.16.255.132, Result: Download unsuccessful
13/04/2024 23:42:08, Type of attack: FTP anon login, Destination: 159.223.49.30
13/04/2024 23:55:44, Type of attack: FTP anon login, Destination: 159.223.49.30
14/04/2024 00:00:36, Type of attack: Brute Force on SSH and Telnet, Destination: 159.223.49.30
14/04/2024 00:06:23, Type of attack: DDOS, Destination: 159.223.49.30
14/04/2024 00:26:03, Type of attack: DDOS, Destination: 159.223.49.30
14/04/2024 00:33:02, Type of attack: DDOS, Destination: 159.223.49.30
14/04/2024 22:03:44, Type of attack: Brute Force on SSH and Telnet, Destination: 159.223.49.30
14/04/2024 22:14:16, Type of attack: Brute Force on SSH and Telnet, Destination: 159.223.49.30
14/04/2024 22:19:59, Type of attack: DDOS, Destination: 159.223.49.30
14/04/2024 22:31:09, Type of attack: FTP anon login, Destination: 159.223.49.30
14/04/2024 22:34:52, Type of attack: FTP anon login, Destination: 159.223.49.30
```

Fig 10 Attack logs in /var/log/attack.log

Attacks carried out by the attacker are being logged in the machine's /var/log/attack.log for tracking purpose by the attacker. Details such as time, type of attack, Destination IP and outcome of attack are being logged into the file. Such information could be useful to the attacker as the attacker could further exploit the vulnerable targets by using more sophisticated attacks.

Discussion

As mentioned earlier, the main objectives of the automation process are:

1. Create a Security Operations Center (SOC) system by installing Elastic Cloud and Honeypot on DigitalOcean to detect and analyze malicious activities on our network
2. Develop pentest scripts to attack the honeypot
3. Monitor the alerts

Objective 1 and 2 were achieved but objective 3 was not met. Achieving Objective 1 and 2 was crucial and also valuable experience as I learnt the process of setting up a SOC system to monitor network activities. Many hours were also being spent in troubleshooting the configurations and this will be valuable experience that we will face in an actual SOC.

Through the project, I also discovered that ELK stack Version 7.17.18 can be quite unstable at times. Many times through the project, the logs does not flow to the correct index in Kibana. For example, sometimes cowrie logs will not appear at the cowrie-logstash index but instead will appear at filebeat index. At times, it will revert back to its index despite no changes being done to the configuration file. This resulted in a lot of time and effort wasted in trying to troubleshoot a false negative problem.

Objective 3 was not achieved as I was not able to configure the required SSL settings for filebeat to connect to Elasticsearch. Visualization was unavailable as well as the logs was behaving erratically.

Through the process of creating a honeypot server, I also discovered that many honeypots are actually outdated in terms of dependencies and also their usage. With the advancement of technology, honeypots might not be so relevant in the current cyber world. More need to be done to detect and observe the behavior of cyber attackers with malicious content. Just like we know that attackers are using fake emails/website to phish for data or conduct cyber attacks, attackers also know that certain servers are fakes and they would not fall into the honeypots as easily as before.

Recommendation

1. Stable ELK/Monitoring system is important
   While setting up the ELK server, I realized that a stable system is very important as it will reduce the amount of time and effort we spend on false negative. E.g. the logs not appearing in the right index and I ended up spending hours trying to resolve it but the logs just suddenly appear in the right places

2. Setting up different security levels according to needs
   There are several security levels we can set while configuring the ELK stack. It ranges from basic security like user login to more secured way using SSL and HTTPS. However, more secured network would required advanced knowledge in setting up the system. Thus it is important to know the level of security required for your server. Basic security can also be setting up firewall rules to restrict access to the system and ports

3. Pentesting and updating system regularly
   Other than updating the system regularly to keep the security system up to date, we also need to conduct pentest on our own servers to check for vulnerabilities. Technology is ever changing and we also need to conduct checks on our server more frequently

Conclusion

In conclusion, SOC is a very important aspect of cybersecurity as we need it to monitor for any malicious activities on the network. Setting up a ELK stack on a server requires a lot of understanding of how the stack works and how each components communicate and connect to each other.

It is also important for us to test out the system we have in our SOC through conducting our own pentesting and monitoring alerts to ensure that the system is functioning as intended.

References

[1] Dejan Tucakov, “**How to Install ELK Stack (Elasticsearch, Logstash, and Kibana) on Ubuntu 18.04 / 20.04**,” Phoenixnap, July 22 2020. [Online]. Available: https://phoenixnap.com/kb/how-to-install-elk-stack-on-ubuntu [Accessed: Feb 25, 2024]

[2 Jeremie Daniel, “**Install And Setup Cowrie Honeypot On Ubuntu(Linux)**,” Medium, Mar 29 2019. [Online]. Available: https://medium.com/@jeremiedaniel48/install-and-setup-cowrie-honeypot-on-ubuntu-linux-5d64552c31dcu [Accessed: Feb 25, 2024]

[3] Valente Security Labs, “**Installing Cowrie Honeypot [Part 8]**,” Medium, May 12 2021. [Online]. Available: https://valentesecuritylabs.medium.com/installing-cowrie-honeypot-part-8-d2924e35727d [Accessed: Feb 29, 2024]

[4] Ekovegeance, “**DDOS**,” Github, Aug 18 2019. [Online]. Available: https://github.com/ekovegeance/DDOS/blob/master/ddos[Accessed: April 2, 2024]

[5] 0xhav0c, “**FTP Pentesting Best Practices**,” Secybr, Aug 22 2022. [Online]. Available: https://secybr.com/posts/ftp-pentesting-best-practices/ [Accessed: April 3, 2024]