

# Penetration Testing Report

For

NBN CORP

05/01/2022

Presented By: Diego Lopez

dtl310@nyu.edu

# Table of Contents

0.0 Executive Summary	3
1.0.0 Introduction	3
1.1.0 Goals and Purpose	3
1.2.0 Rules of Engagement	4
1.3.0 Scope	4
2.0.0 Methodology	4
2.1.0 Risk Calculation	5
2.2.0 Scanning	5
2.3.0 FTP Testing	5
2.4.0 Web Application Testing	5
2.5.0 Brute Force SSH Scanning	6
2.6.0 Loot and Privilege Escalation	6
3.0.0 Findings	6
3.1.0 Anonymous Credentials FTP Login	6
3.2.0 Reflected Cross Site Scripting (XSS)	7
3.3.0 Path Traversal	8
3.4.0 Sensitive Information Disclosure	9
3.5.0 Command Injection Attack	10
3.6.0 SQL Injection Attacks	11
3.7.0 SSH Brute Force Attack	12
4.0 Conclusion	12
4.1 Next Steps and Immediate Action	13
5.0.0 Appendix	13
5.1.0 Flags	13
5.2.0 Tool Output	15
5.3.0 Terminal Shells Access	16
5.4.0 Web Application Vulnerability Screenshots	18
5.5.0 Sensitive Information Disclosure	20

# 0.0 Executive Summary

A penetration test was performed on the provided NBN images, a NBN Server instance and a NBN Client instance.

The NBN web server had numerous potentially harmful vulnerabilities, exposing sensitive company and customer data, server information, and potential harm to visitors of the site. Direct terminal access to the NBN server was gained. Direct terminal access to the NBN client was gained.

Few security protocols and best practices were enforced, however when done correctly, were very potent and effective. In its current state, **we do not recommend launching the production server until the critical vulnerabilities can be addressed**, as detailed in the Findings section.

The most concerning vulnerabilities were the following:

- Brute Force SSH - High - Due to various exposures of usernames on the server, as well as passwords being compromised as part of the *RockYou* database, terminal access on both the NBN Server and NBN Client were gained and pose a direct threat to company assets.
- Command Injection - High - Due to a comment left in the production server HTML page after customer registration, information leading to a command injection attack is revealed and was validating through testing. Document writing for customer registration involves directly using system commands, which may allow a potential attacker to inject malicious commands under privilege directly to the shell.

## 1.0.0 Introduction

NBN Corp (NBN), formerly known as Network Broadcast News, Net Broadcast Network, and Near-Earth Broadcast Network, is a Los Angeles based media conglomerate which has just suffered a major cyber security breach. This breach resulted in loss of customer and employee data which originated from their external facing servers.

Due to this recent breach, NBN has sought out a team to perform a penetration test in order to better their security posture and discover potential remaining vulnerabilities. An independent contractor, Diego Lopez, has been tasked with performing this test.

### 1.1.0 Goals and Purpose

The main objective of the test is to discover, enumerate and penetrate the systems provided by NBN for testing. The type of test performed was a **black box test** (Red Team). This test was performed to mimic the capabilities of future attackers seeking to exploit NBN systems. No information on the system or infrastructure was provided and all findings were discovered in manners that potential attackers could emulate. As such, findings should be taken **extremely seriously** in relation to security of NBN systems.

## 1.2.0 Rules of Engagement

Specific rules of engagement (ROE) were defined for the test and are enumerated below.

### 1.2.1 Testing Schedule

The test will be conducted from 04/25/2022 - 05/13/2022. Network testing will be performed throughout the day, however the majority will be done outside business hours (After 5PM EST). No briefings will be conducted for this test as requested by NBN. An extensive report of findings will be delivered by 05/13/2022 to NBN as per agreement.

### 1.2.2 Point of Contact

The main point of contact for the testing team will be Mr. Lopez Directly. He is reachable by email at [diego.lopez@nyu.edu](mailto:diego.lopez@nyu.edu). His Network ID (NetID) for NYU is **dtl310**. The main point of contact for NBN will be CEO Gibson.

### 1.2.3 Disclosure of Sensitive Information

All systems provided for testing by NBN were virtualized on a known, private, residential network. Although the testing team would have preferred to virtualize the testing images under a host-only network configuration, or a separate NAT Network, this configuration was attempted and resulted in network connectivity issues. As such, the images were virtualized under a NAT-Bridged configuration directly to the network. Although not ideal, we believe the risk posed by this change is minimal given the network is private and no intrusions have been recorded, to date.

All sensitive data retrieved from testing, including customer information, employee information, vulnerability enumeration, etc were stored on a virtualized machine on an encrypted hard drive. At the conclusion of the test, and at the permission of NBN, all information collected will be destroyed.

## 1.3.0 Scope

The target scope is specifically 2 disk images that have been provided by NBN.

- Web Server instance
- Client instance

We will test the machines **over the network**, as a potential attack may. We will not change system configurations or install new software onto the targets.

Denial of Service (DoS) attacks will **not** be attempted.

## 2.0.0 Methodology

A detailed description of our methodology and approach is detailed below.

## 2.1.0 Risk Scoring and Evaluation

The risk categories for vulnerabilities discovered are detailed below:

Risk Category	Rationale	Equivalence to CVSS™
High	These vulnerabilities are readily exploitable and pose significant risk to NBN and its employees	7.0 - 10.0
Medium	These vulnerabilities are possible to exploit, but are either difficult or do not pose a substantial risk to NBN	5.0 - 7.0
Low	These vulnerabilities are exceedingly difficult to exploit, or result in a very minimal increase in attack surface	2.0 - 5.0
Information	These vulnerabilities result in disclosure of potentially sensitive information that may aid an attacker	0.0 - 2.0

## 2.2.0 Scanning

Vulnerability scans were performed on the server instance using *nmap*, a multi-purpose network scanner. These included all-port scans, *badsum* scans, and version scans (Appendix 5.2.0). The open ports and services running were enumerated and investigated individually for potential CWE's and other known vulnerabilities in version.

## 2.3.0 FTP Testing

An open FTP server was discovered on the server, and was investigated. Various spurious credentials were attempted to authenticate, including anonymous credentials, which were accepted (detailed more in findings). The available directories were traversed and **flag 3 was discovered** (Appendix 5.1.0).

## 2.4.0 Web Application Testing

2 open web servers were discovered on the server instance and were investigated. Attack proxies (Burp and ZAP) were initialized. The 2 web servers were initially crawled, and then scanned for vulnerabilities (5.2.0). Automated scans were performed using ZAP. Potential vulnerabilities were identified, enumerated and manually investigated for potential impact.

These included reflected and stored XSS, path traversal, remote OS execution, SQLi, and sensitive disclosure of information (discussed in 3.0.0 Findings).

## 2.5.0 Brute Force SSH Scanning

Due to a path traversal vulnerability in the web application, `/etc/passwd` was viewed and a possible username, *gibson*, was recorded. Utilizing metasploit's brute force SSH login scanner, along with *rockyou.txt* as a password list, we were able to identify *gibson*'s password on the server and **gain terminal access to the NBN server** (Appendix 5.3.0). For informational purposes only, *gibson*'s password was "digital".

## 2.6.0 Loot and Privilege Escalation

After successfully logging in with *gibson*'s credentials onto the server, various system directories were searched for potential sensitive information. *Gibson*'s credentials were also found to authenticate the web server, where **flag 2** was found (Appendix 5.1.0).

Attempts were made to search for files containing the phrase "password", and were manually explored. The kernel version was recorded at 4.15.0-47-generic, and exploits were searched for using CVE, metasploit, and exploitDB. Backup files were found in `/var/backups`, however permission was not granted to view these from *gibson*'s credentials.

Services running as root were examined. The MariaDB instance was connected to (using credentials *root:digital*), and the user table was examined. The password hash for user *stephenson* was extracted, and cracked using *hashcat*. For informational purposes, his password was "pizzadeliver" (Appendix 5.3.0).

SSH attempts to log into the NBN Client machine were made, and *stephenson*'s credentials were successful. We gained **terminal access to the NBN client instance** (Appendix 5.3.0).

## 3.0.0 Findings

Below are enumerated findings, as well as possible risk posed to NBN and recommendations for mediation.

### 3.1.0 Anonymous Credentials FTP Server Login

#### 3.1.1 Background

File Transfer Protocol (FTP) servers are common services to transfer files from a server to a client over a network. FTP servers are usually secured with authentication, requiring a valid username and password combination to access given resources on a server. One possible configuration that a server may allow are *anonymous credentials*, essentially allowing anyone with network access to the server to access resources.

### 3.1.2 Details

The NBN server authenticated with anonymous credentials, allowing access to company resources. This resulted in disclosure of **flag 3**, which was accessed through the *gibson* directory (Appendix 5.1.0).

### 3.1.3 Risk Analysis

Risk Category	Low
Explanation	Anonymous credentials expose potentially sensitive information that may aid a future attacker. These may include private SSH keys, command history, kernel version, users, etc. Although, the version of this FTP service does not have many easily exploitable vulnerabilities.

### 3.1.4 Recommendations

We recommend disabling anonymous access to the FTP server, requiring company credentials to access critical resources. This represents a somewhat trivial fix, as anybody accessing company resources who is authorized to do so should register an account with IT for proper classification and identification.

## 3.2.0 Reflected Cross Site Scripting (XSS)

### 3.2.1 Background

Cross Site Scripting (XSS) attacks are a type of injection attack where a malicious actor can inject scripts into otherwise trusted and reputable sites, resulting in potential risks towards future users (OWASP). These attacks generally occur due to a lack of sanitization of inputs, where form inputs may be read directly without escaping.

Reflected XSS, or “non persistent” XSS occurs when the payload is delivered and executed via a single request primarily through crafted URL parameters (OWASP Reflected XSS).

### 3.2.2 Details

The production web server login was vulnerable to reflected XSS attacks (Appendix 5.4.0). Although attempts were made to sanitize inputs for SQL injection (SQLi) attacks for authentication of employee credentials, our attempts to perform reflected XSS attacks were successful as login attempts displayed dynamic text in response to parameters passed to the form inputs.

Javascript was executed using the form input for “username” login (Appendix C). This poses a potential threat to future *employee* visitors to the site, as this is an employee login page. It may be possible for an attacker to execute malicious code and otherwise target NBN staff.

### 3.2.3 Risk Analysis

Risk Category	Medium
Explanation	Reflected XSS poses a risk to potential visitors of the website. Although this alone would be considered somewhat “low” risk, this page specifically affects a employee login page, which increases the risk for NBN directly.

### 3.2.4 Recommendations

We recommend sanitizing form inputs for XSS attacks, which would help in mitigating the vulnerability. For even better security, we recommend **not displaying** dynamic text on login failure, as this discloses potential information about the server that is not necessary for the employee and serves to expand the information an attacker can gain from the system. Specific implementation details can be found in the OWASP XSS Prevention cheat sheet below.

### 3.2.5 References

[1]

<https://owasp.org/www-community/attacks/xss/>

[2]

[https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/01-Testing\\_for\\_Reflected\\_Cross\\_Site\\_Scripting.html](https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting.html)

[3]

[https://cheatsheetseries.owasp.org/cheatsheets/Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)

## 3.3.0 Path Traversal

### 3.3.1 Background

Path traversal attacks occur when sensitive data stored **outside** the web root folder is accessed through unauthorized means. This occurs mostly through the manipulation of the URL passed to the web server through character sequences such as “../”.

### 3.3.2 Details

The staging server was vulnerable to a path traversal attack. This resulted in the disclosure of the /etc/passwd file, revealing highly sensitive information such as running services, account names, etc (Appendix 5.4.0). The URL provided was appended was

[:?authenticated=1&list=%2Fetc%2Fpasswd](http://localhost:8080/?authenticated=1&list=%2Fetc%2Fpasswd)

Importantly, this discloses user account information that may be used by attackers for brute force attacks on SSH, direct login, and web login (discussed below).



### 3.3.3 Risk Analysis

Risk Category	Information
Explanation	Given the direct exposure of /etc/passwd and running services and accounts, this dramatically increases the information an attacker may have towards the system. This directly leads to possible brute force SSH attacks, or brute force web login attempts.

### 3.3.4 Recommendations

Disabling external access to the staging server would be the most effective solution, as the “test” authorization allows for this to occur in the staging server. As a note, this attack was attempted on the production server as well, however authorization was not provided and such the attack was mediated.

As to specifically preventing this attack, OWASP recommends ensuring that a user cannot specify an entire path, and that any path provided by users be wrapped with server provided path code (OWASP).

### 3.3.5 References

[1] [https://owasp.org/www-community/attacks/Path\\_Traversal](https://owasp.org/www-community/attacks/Path_Traversal)

## 3.4.0 Sensitive Information Disclosure

### 3.4.1 Background

When users make GET requests to a web server, it is possible that in the web page returned by the server, sensitive information regarding the server itself may be left in the source code for the page that may not have been intended for use by the client. This may include comments, possible directories, mechanisms of action, etc.

### 3.4.2 Details

In the web page code returned after using the “Email Registration” form, code is visible disclosing the existence of the “/data” directory, as well as leading information on potential command injection (Appendix 5.5.0).

In this data directory, we find **flag 1** (Appendix 5.1.0) as well as a customers list (Appendix 5.5.0). The customer list displays emails and names of people who have registered using the form.

### 3.4.3 Risk Analysis

Risk Category	Information
Explanation	Information regarding a potential command injection exploit was sent to visitors of the site in the comments of the web page. This could directly aid an attacker in exploiting the server. Sensitive customer and company data was also exposed in the form of email registration lists as well as a flag.

### 3.4.4 Recommendations

We recommend that sensitive data be stored outside the web root directory. This would provide additional security for sensitive files and possibly mediate these attacks.

## 3.5.0 Command Injection Attack

### 3.5.1 Background

Command injection attacks occur when a vulnerable application passes user data such as cookies, http headers, or forms directly to a shell on the system for manipulation or storage (OWASP). These commands are then executed with the privilege of the application, which may lead to potential attacks or control of the system.

### 3.5.2 Details

As seen in Appendix 5.5.0, the email and username are passed to a command shell which echos them to a file. The production web server is vulnerable to command injection attacks by crafting form input that is able to escape the string and insert additional commands to the shell. This is especially concerning due to the elevated privileges that a web application runs system commands with.

### 3.5.3 Risk Analysis

Risk Category	High
Explanation	This vulnerability allows for direct injection of commands to the system, potentially exposing remote code execution and shell access for attackers.

### 3.5.4 Recommendations

To mediate these attacks, we recommend removing the comment with the debug command that displays how the webpage works to users of the site. This would obfuscate the mechanism by which data is stored, and potentially delay an attack.

Furthermore to mitigate the risk further, we recommend sanitizing all inputs that are passed to shells directly, and possibly use a different mechanism to write to files that do not execute commands directly under the privilege of the web server.

### 3.5.5 References

[1]

[https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

## 3.6.0 SQL Injection Attacks

### 3.6.1 Background

SQLi attacks occur when the server executes a SQL query based on user input without sanitization. A potential attacker is able to modify the structure of the original query to view sensitive information about the server and potentially modify data (OWASP).

### 3.6.2 Details

The staging server was found to be vulnerable to SQLi attacks, as well as displaying an error page to the user upon a broken query that reveals sensitive information about the server. We note that the production server had proper sanitization elements in place for SQLi (Appendix 5.4.0).

### 3.6.3 Risk Analysis

Risk Category	Medium
Explanation	The staging server login page produced an open error page that displayed critical information about the server, including the query string that could be exploited by a potential attacker. No sanitization was done on the user inputs, making this vulnerability viable for potential attackers to exploit.

### 3.6.4 Recommendations

We recommend that inputs to SQL queries be sanitized, as they are in the production server. Quotations, special characters, and comments should be escaped before sent through an SQL query to the database. More information can be found in the references at OWASP SQLi prevention cheat sheet (OWASP).

### 3.6.5 References

[1]

[https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)

[2]

[https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)

## 3.7.0 SSH Brute Force Attack

### 3.7.1 Background

Brute Force attacks occur when a potential username is exposed to an attacker, and the password space is either known or small. Using automated tools, an attacker may provide a wordlist (password space) to explore for a given username and perform a grid search over all passwords in attempts to successfully authenticate.

### 3.7.2 Details

Given the username extracted from the path traversal exploit, as well as the password list *rockyou.txt*, a brute force SSH attack was performed using Metasploit's auxiliary module **ssh\_login**. A password combination was found as "gibson:digital", and successfully authenticated to the server. We gained **terminal access to the NBN server** (Appendix 5.3.0).

### 3.7.3 Risk Analysis

Risk Category	High
Explanation	Given the username <i>gibson</i> exposed by the path traversal vulnerability, as well as the password being present in our wordlist <i>rockyou.txt</i> , SSH brute force poses a serious threat to NBN as we were able to gain direct terminal access to the server.

### 3.7.4 Recommendation

We recommend *gibson* change their password, as this password is present in the *RockYou* database leaked wordlist. Having a strong password is essential to security, **especially when this profile is available for SSH to key NBN systems**. We also recommend that the path traversal vulnerability and the FTP vulnerability be patched immediately as these directly lead to information that can be used in a SSH brute force attack on the server.

## 4.0 Conclusion

The goal of this test was to explore the vulnerability space of the 2 instances provided by NBN: A server instance, and a client instance. Numerous vulnerabilities were identified and exploited, leading to direct terminal access to both instances. Privilege escalation was explored on both machines, however we were unable to achieve root on these systems. Some promising leads, such as *tee* running as sudo and not requiring password, were explored, however we were

unable to achieve root. Attempts were made using various kernel exploit modules, however these were unsuccessful. Possible future avenues for achieving root would be as follows:

- Utilizing *tee* to inject code into a privileged script to add *gibson* to */etc/sudoers*
- Utilizing the previously found command injection to add *gibson* to */etc/sudoers* under privilege of the web server
- Exploring possible application vulnerabilities on both the client and the server instance

Time proved to be a constraint. Given more time, the team believes that we would be able to achieve root, and thus consider these potential vulnerabilities to be of high importance alongside the discovered web application vulnerabilities.

## 4.1 Next Steps and Immediate Action

Immediate action can and should be taken to address critical vulnerabilities in NBN instances. These are enumerated below:

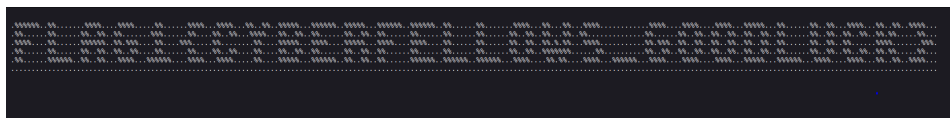
- Change *gibson* and *stephenson*'s passwords - both of these employee passwords were found to be part of the *RockYou* leaked database, and such are available to potential attackers. This poses significant risk and may be mitigated immediately.
- Add sanitization for XSS to the production server login - Although this login provides sanitization to SQLi, there appears to be no sanitization of form inputs for XSS, and as such is vulnerable to such attacks. Adding this would drastically improve security posture for NBN as this login page specifically targets employees.
- Disable anonymous login to FTP - Anonymous credentials may be supplied to access critical company information, specifically flag 3. Disallowing anonymous credentials would restrict access to only authenticated users, and significantly improve security.
- Add sanitization for Command Injection - Customer information is saved to a file through the direct use of system commands, which potentially allows attackers to alter the command through careful syntax formatting and run system level commands as the web server. By adding input sanitization, or using an alternative form of write to avoid direct system commands, this vulnerability may be mitigated.

## 5.0 Appendix

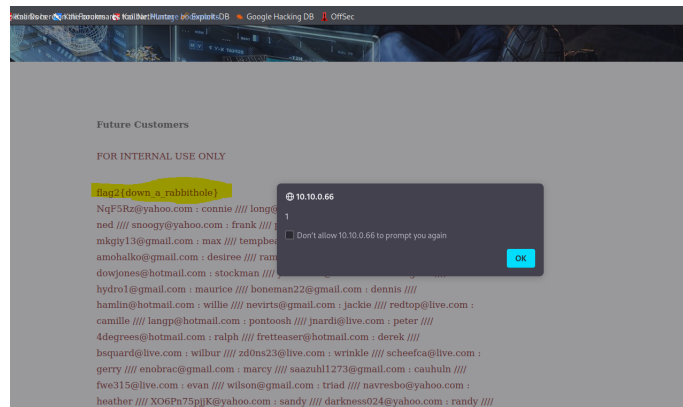
### 5.1.0 Flags

Various flags were found on the machines. They are listed below in order:

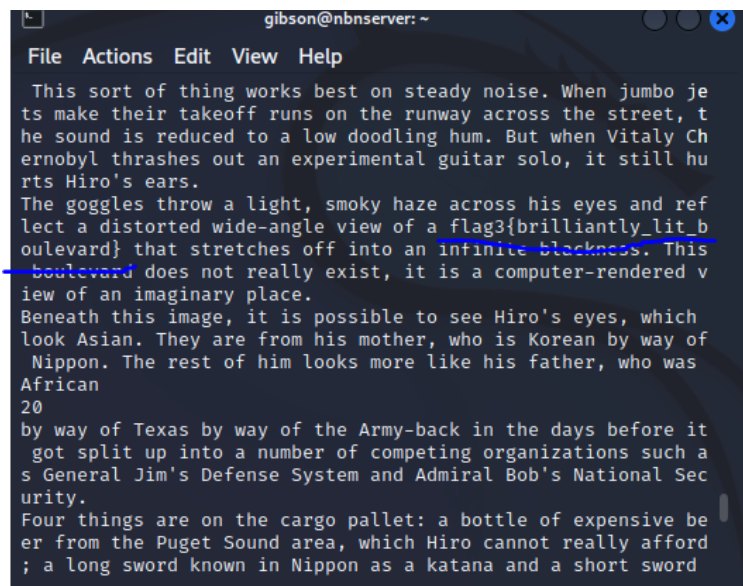
#### 5.1.1 Flag 1



### 5.1.2 Flag 2



### 5.1.3 Flag 3



## 5.2.0 Tool Output

### 5.2.1 Nmap Automated Scan

#### Scan Summary

Nmap 7.92 was initiated at Mon May 9 09:47:14 2022 with these arguments:

nmap -iC -iV -p- -Pn -oX initialscan.xml 10.10.0.66

Verbosity: 0; Debug level: 0

Nmap done at Mon May 9 09:47:42 2022; 1 IP address (1 host up) scanned in 28.58 seconds

#### 10.10.0.66

##### Address

- 10.10.0.66 (ipv4)

##### Ports

The 65531 ports scanned but not shown below are in state: **closed**

- 65531 ports replied with: **conn-refused**

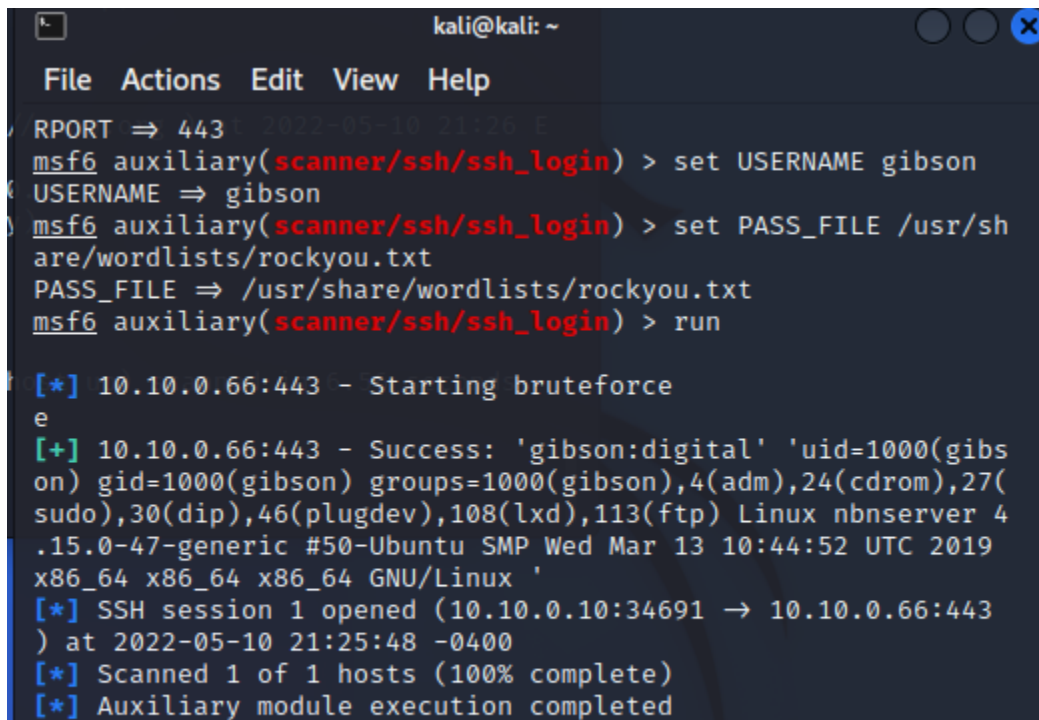
Port	tcp	State (toggle closed [X]   filtered [F])	Service	Reason	Product	Version	Extra info
80	http	open	http	syn-ack	Apache httpd	2.4.29	(Ubuntu)
	http-site		nginx				
	http-robots.txt						
	http-server-header						
443	ssl	open	ssh	syn-ack	OpenSSH	7.6p1 Ubuntu amd64	Ubuntu Linux, protocol 2.0
	ssh-hostkey						
	ssh-hostkey						
8001	tcp	open	http	syn-ack	Apache httpd	2.4.29	(Ubuntu)
	http-server-header						
	http-site						
	http-robots.txt						
85534	tcp	open	ftp	syn-ack	vftpd	3.0.3	
	ftp-epsd						
	ftp-epsd						

### 5.2.2 ZAP Automated Active Scan Output

Active (12)
✖ Cross Site Scripting (Persistent) (2)
GET: http://10.10.0.66/data/customer.list
GET: http://10.10.0.66/data/customer.list
✖ Cross Site Scripting (Reflected) (2)
GET: http://10.10.0.66:8001/login.php?login=Enter&password=ZAP&username=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E
GET: http://10.10.0.66:8001/login.php?login=Enter&password=ZAP&username=%3C%2Fheader%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cheader%3E
✖ Path Traversal
GET: http://10.10.0.66:8001/internal/customers.php?authenticated=1&list=%2Fetc%2Fpasswd
✖ Remote OS Command Injection (2)
GET: http://10.10.0.66/email=foo-bar%40example.com%27%26sleep+15%26%27&name=ZAP%2F
GET: http://10.10.0.66/email=foo-bar%40example.com&name=ZAP%2F%27%26sleep+15%26%27
✖ SQL Injection
GET: http://10.10.0.66:8001/login.php?login=Enter&password=ZAP&username=ZAP%27+AND+1%271%27%3D%271%27+--
✖ Application Error Disclosure (154)
✖ Directory Browsing (17)
✖ X-Frame-Options Header Not Set (184)

## 5.3.0 Terminal Shells Access

### 5.3.1 SSH Brute Force using Msfconsole

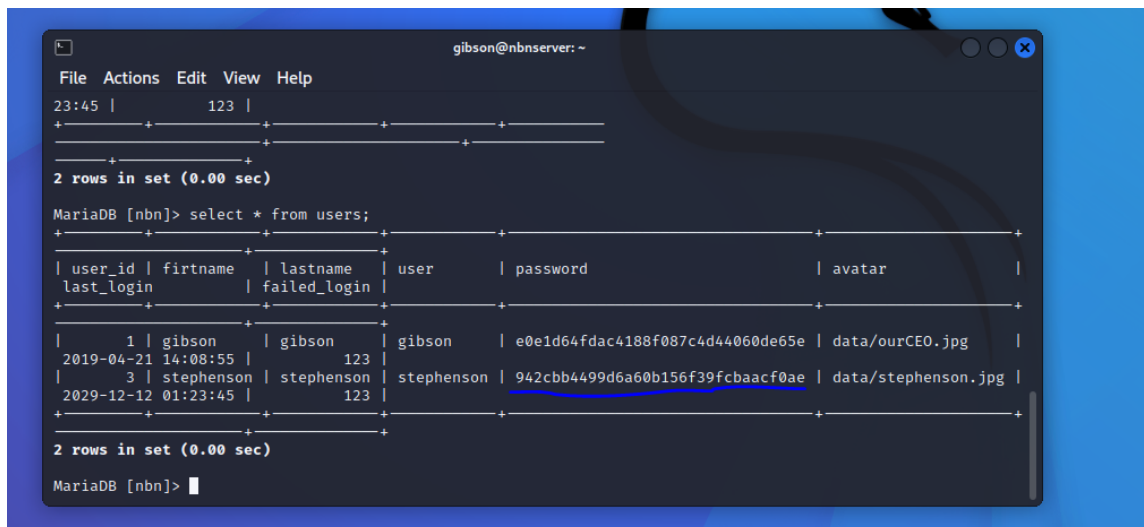


```
kali@kali: ~  
File Actions Edit View Help  
RPORT => 443 2022-05-10 21:26 E  
msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME gibson  
USERNAME => gibson  
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt  
PASS_FILE => /usr/share/wordlists/rockyou.txt  
msf6 auxiliary(scanner/ssh/ssh_login) > run  
[*] 10.10.0.66:443 - Starting bruteforce  
e  
[+] 10.10.0.66:443 - Success: 'gibson:digital' 'uid=1000(gibson) gid=1000(gibson) groups=1000(gibson),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd),113(ftp) Linux nbnsnserver 4.15.0-47-generic #50-Ubuntu SMP Wed Mar 13 10:44:52 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux'  
[*] SSH session 1 opened (10.10.0.10:34691 -> 10.10.0.66:443) at 2022-05-10 21:25:48 -0400  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed
```

### 5.3.2.0 Password Cracking using Hashcat

#### 5.3.2.1 Password hash in MariaDB

A password hash for user *stephenson* was found after logging into the MariaDB, and connecting to the database.



```
gibson@nbnsnserver: ~  
File Actions Edit View Help  
23:45 | 123 |  
+-----+-----+-----+-----+  
+-----+  
2 rows in set (0.00 sec)  
MariaDB [nbn]> select * from users;  
+-----+-----+-----+-----+  
| user_id | firstname | lastname | user | password | avatar |  
+-----+-----+-----+-----+  
| 1 | gibson | gibson | gibson | e0e1d64fdac4188f087c4d44060de65e | data/ourCEO.jpg |  
| 2019-04-21 14:08:55 | 123 |  
| 3 | stephenson | stephenson | stephenson | 942cbb4499d6a60b156f39fcbacaf0ae | data/stephenson.jpg |  
| 2029-12-12 01:23:45 | 123 |  
+-----+-----+-----+-----+  
2 rows in set (0.00 sec)  
MariaDB [nbn]>
```



### 5.3.2.2 Password Cracking using Hashcat Output

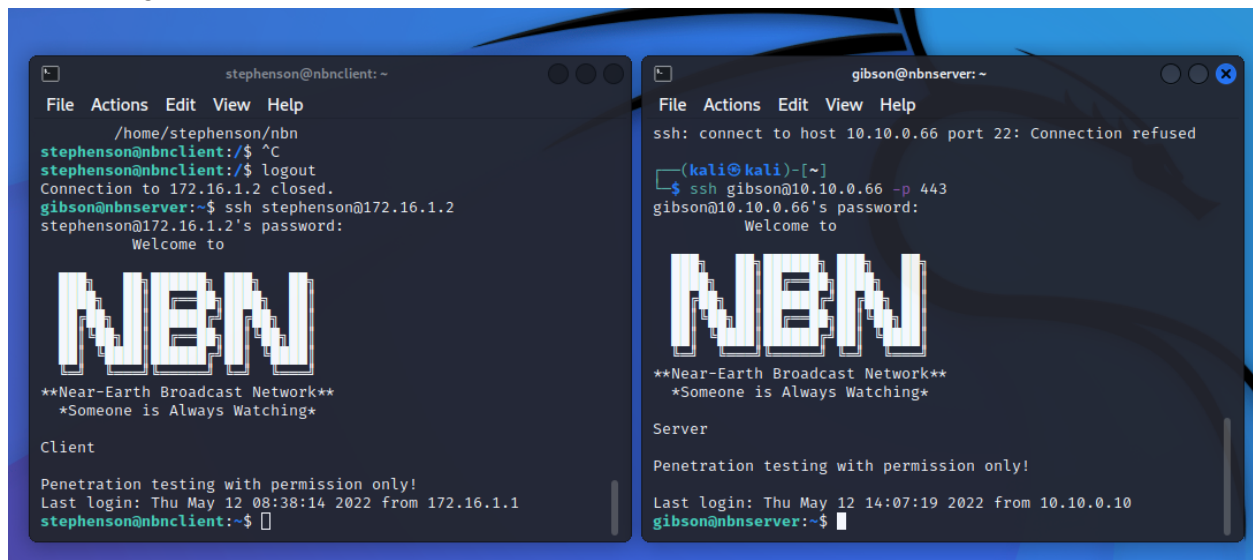
We used hashcat to crack *stephenson*'s password. We specified *md5* as the algorithm and *rockyou.txt* as the wordlist.

```
Dictionary cache building /usr/share/wordlists/rockyou.txt: 3
Dictionary cache building /usr/share/wordlists/rockyou.txt: 6
Dictionary cache built:
* Filename .. : /usr/share/wordlists/rockyou.txt
* Passwords.. : 14344392
* Bytes.....: 139921507
* Keyspace.. : 14344385
* Runtime ... : 1 sec

942cbb4499d6a60b156f39fcbacfc0ae:pizzadeliver
```

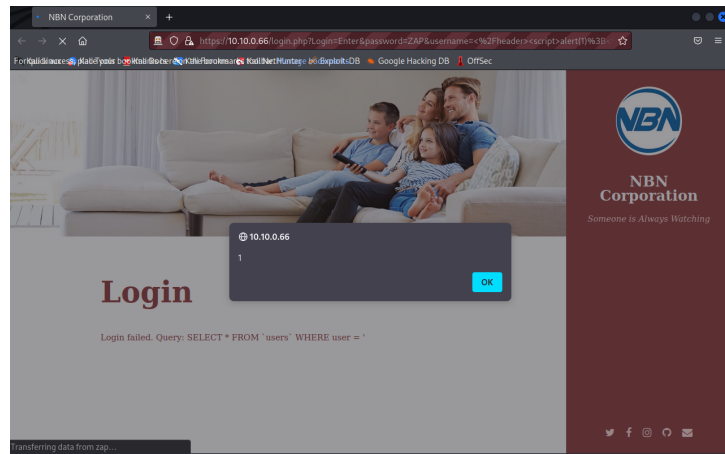
### 5.3.2 Terminal Access Proof

Using credentials *gibson:digital* for the NBN Server and *stephenson:pizzadeliver* for the NBN Client, we gained direct terminal access.



### 5.4.0 Web Application Vulnerability Screenshots

### 5.4.1 Reflected XSS on Login Page



### 5.4.2 Path Traversal

Using the path traversal exploit, we were able to view the contents of `/etc/passwd`.

FOR INTERNAL USE ONLY

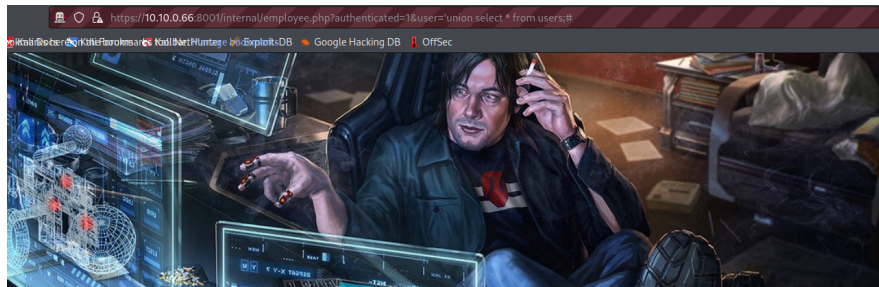
```

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr
/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool
/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr
/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailin
g List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin
/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin
/nologin syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin _apt:x:104:65534::/nonexistent:
/usr/sbin/nologin lxd:x:105:65534::/var/lib/lxd/:/bin/false uuid:x:106:110::/run/uuid:
/usr/sbin/nologin dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin pollinate:x:109:1::/var/cache
/pollinate:/bin/false sshd:x:110:65534::/run/ssh:/usr/sbin/nologin
gibson:x:1000:1000:gibson:/home/gibson:/bin/bash ftp:x:111:113:ftp daemon,,,:/srv
/ftp:/usr/sbin/nologin mysql:x:112:115:MySQL Server,,,:/nonexistent:/bin/false

```

### 5.4.3.0 SQLi Attack Staging Server

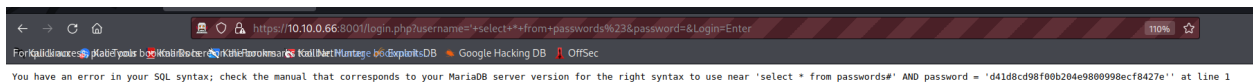
#### 5.4.3.1 Breaking login through SQL query formatting



Welcome, 'union select \* from users;

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.

#### 5.4.3.2 SQL error page



## 5.5.0 Sensitive Information Disclosure

### 5.5.1 Command Injection Comment

```
view-source:https://10.10.0.66/?name=555&email=#
14 <div id="wrapper">
15
16 <!-- Main -->
17 <div id="main">
18
19 <!-- One -->
20 <section id="one">
21 <div class="image main" data-position="center">
22 
23 </div>
24 <div class="container">
25 <header class="major">
26 <p>Thanks for Registering!<!--DEBUG
27 $cmd = shell_exec("echo ' ' . $_GET['email'] . " : " . $_GET['name'] . " ' >> /var/www/html/data/customer.list " );
28 --></p>
29
30 <h2>Near-Earth Broadcast News</h2>
31 <p>Connecting You to the World</br>
32 </p>
33 </header>
34 <p>The largest media conglomerate in the world, NBN operates five of the six top-rated content :
35
36 <p>We do more than simply read the communication and entertainment market; we steer it.</p>
37 </div>
38 </section>
```

### 5.5.2 Sensitive Customer List

```
NqFSRz@yahoo.com : connie ////
long@gmail.com : capone ////
hjk12345@hotmail.com : red ////
snoogy@yahoo.com : frank ////
polobear@yahoo.com : jess ////
mkgiyl3@gmail.com : max ////
tembeauties@live.com : peterpiper ////
amohalko@gmail.com : desiree ////
ramy43@gmail.com : greatone ////
dowjones@hotmail.com : stocknan ////
yahotmail@hotmail.com : eugene ////
hydrol@gmail.com : maurice ////
boneman22@gmail.com : dennis ////
hamlin@hotmail.com : willie ////
nevirt@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 ////
zdmns2@live.com : wrinkle ////
scheefca@live.com : gerry ////
enobrac@gmail.com : marcy ////
saazuhi1273@gmail.com : cauhtn ////
faw215@live.com : evan ////
wilson@gmail.com : triad ////
navresbo@yahoo.com : heather ////
X06Ph75pjk@yahoo.com : sandy ////
darkness024@yahoo.com : randy ////
jjstrokes@live.com : beansko ////
zimago@yahoo.com : george ////
katrina@gmail.com : harald ////
awesone@gmail.com : larry ////
jess@yahoo.com : jesse ////
<script>alert("Vulnerable")</script> ////
fje.edf : <script>alert("Vulnerable")</script> ////
: or 1 = 1 ////
foo-bar@example.com : ZAP ////
foo-bar@example.com : ZAP ////
foo-bar@example.com : ZAP ////
c:/windows/system.ini : ZAP ////
c:/windows/system.ini : ZAP ////
c:/windows/system.ini : ZAP ////
/etc/passwd : ZAP ////
/etc/passwd : ZAP ////
c/ : ZAP ////
/ : ZAP ////
c:\ : ZAP ////
```