Penetration Test Report

Near-Earth Broadcast Network

Alexander Lotero
CS-GY 6903
Professor Jared Smith
May, 2022

Apex Security Consulting

1439 S. Point View St.
Suite 6
Los Angeles, CA 90035
United States of America

Telephone: 678-999-8212
Email: info@apexsec.org
Web: http://www.apexsec.org

May, 2022

Bill Gibson, CISO
Near-Earth Broadcast Network
NBN Corp
1800 Archer Street
New York, NY

A. Executive Summary

a. Purpose

Near-Earth Broadcast Network (NBN) contracted Apex Security Consulting to perform a penetration test to determine their systems' vulnerability to outside threats. All testing activities were performed with the intention of simulating an external threat, meaning, no prior knowledge of NBN's internal systems, no preexisting backdoor into NBN's network (e.g. remote access enabled for a privileged employee), and no detailed description of NBN's infrastructure. The goal of Apex's testing was to:

- Locate a vulnerability that enables external users to gain unintended access to internal systems
- Rank these vulnerabilities by their ease-of-exploit and the resulting exposure of company assets
- Report patching options to harden the system and prevent these vulnerabilities from being exploited

These tests were done with no additional privileges to the average user interacting with NBN's public-facing web page. All testing was performed under safe conditions on virtual machine images representative of NBN's live network.

b. Major flaws

This report will outline numerous vulnerabilities located during our testing, but each of these vulnerabilities was made possible by two important flaws in the NBN's web server, specifically the employee login page. Firstly, within the login page, in addition to a status message, the web server is also writing to the screen the structured query language (SQL) command that was used to test the user-provided credentials. This will be explained in greater depth in a later section of the report, but to summarize, it provides users, and in turn malicious actors, with useful insight into how login attempts are handled. This insight helps attackers formulate inputs that exploit the login page without ever needing valid credentials. Second, the employee credentials themselves do not appear to be held to any particularly secure standard. Specifically, the lack of requirements for complex passwords on employee accounts enabled us as attackers to "guess" our contact, Bill Gibson's password in less than a minute. These two vulnerabilities provided us as attackers with unintended access to NBN's systems and data.

c. Fixes to major flaws

To address these weaknesses, we highly recommend three immediate actions. First, the personnel in charge of development and maintenance of NBN's website must disable the detailed output of a failed login attempt to the user and only output the message itself. In its current state, the page will output "Login failed," followed by the SQL command used to query the database. The more secure method should only output the "Login failed" message. A more detailed message is optional (e.g. "Login failed, invalid credentials"), but not required. Second, a system should be implemented to require complex passwords for employees such as the PassRequirments jQuery plugin [1]. This system will create a stricter minimum for password complexity that will help ensure that attackers must invest more than a few minutes into guessing valid credentials. Finally, alerts should be enabled to notify proper personnel when a user has logged in remotely. This will provide NBN with the ability to review these logins, and if they are determined to be malicious, terminate this user's access before further assets are compromised.

d. Security score

After careful consideration of the test results, each individual vulnerability has been given a risk score according to its likelihood and severity/consequence using a risk scoring matrix by project-management.com [2]. These individual scores were compiled and averaged to create NBN's overall risk score of 14 (high). The risk matrix used has four score levels: low, medium, high, and extreme. Our penetration test scored NBN within the high risk category, just below extreme. The scoring process will be explained in greater detail in a later section.

Table of Contents

1. Introduction

As alluded to within the executive summary, the testing was performed within the black-box, red team methodology. This means our team was not provided with any additional insight about Near-Earth Broadcasting Network's (NBN's) systems and infrastructure beyond what is publicly available to the average user. The goal of this testing methodology is to provide an authentic simulation of an outside threat actor.

For this test in particular, the goals were as follows:

- Identify vulnerabilities within NBN's public facing web server that allows unintended behavior of the web page or server,
- Gain access to internal systems via the public facing web page,
- Formulate a list of vulnerabilities on the web page, the web server, and the internal NBN host and a list of patching options to address these vulnerabilities

Beyond just identifying vulnerable points of access, the NBN test environment, made up of two virtual machine images, was populated with capture the flag style "flag" representative of important files or sensitive company information. These flags were included to serve as a proof-of-exploitation. Each flag discovered and/or retrieved demonstrates that the vulnerability by which they were discovered has the potential to expose company secrets and sensitive data. Two such flags were retrieved during our testing.

Apex Security Consulting's (ASC's) approach to this test can be summarized by two steps:

1. Gain access to the network via the publicly available NBN web page, and
2. Once inside the network, discover what avenues exist to tamper with, steal, and destroy data.

Each of these two steps summarizes roughly a week of the agreed upon two week penetration testing period, respectively. The first half of the testing period was focused on analyzing the public web server to discover/create a point of entry to NBN's network. The second half saw our team answer the following question: "an attacker has gained a presence on the internal network, now what can they do to further exploit these systems?"

Before beginning the test itself, ASC and ASC's NBN contact, Bill Gibson, worked closely to clearly define the scope for this process. In this case, NBN had a somewhat broad goal for this test: how is our network, consisting of a web server and employee hosts, vulnerable to outside threat actors. Luckily, to NBN's credit, they met their broad goals with a mostly unrestricted scope. The only restrictions outlined within the scope were two-fold: no distributed denial of service attacks[1] and no actions that imply physical access to the organization's machines. As such, our responsibilities as the penetration testers were clear. Upon the conclusion of the two week testing period, we must provide NBN with a list of vulnerabilities discovered, detailed explanations of how these vulnerabilities were identified, what implications they have and how severe is the risk that they present, and finally, how can these risks be addressed to ensure a more secure environment for public deployment.

Finally, we would like to re-enforce the overall score that NBN was given on their cyber security risk stature, 14 (high). This score was calculated by averaging the individual risk score given to the four identified vulnerabilities. Each vulnerability is scored by likelihood and consequence of occurrence, more on the individual scores later. These scores are then added together and divided by four (the number of recorded vulnerabilities). These calculations left us with a score of 14 out of 25. The risk matrix used[2] has a minimum score of 1 and a maximum score of 25. The scores are broken down into labels roughly as follows:

- 1-3: low risk
- 4-7: medium risk
- 8-15: high risk
- 16-25: extreme risk

NBN received a score of 14 (high), meaning the organization is operating with what most organizations would agree is an unsafe amount of risk. We highly recommend NBN incorporate each of the fixes outlined later in our findings section.

[1] Distributed Denial of Service attacks (DDoS) are attacks in which resources, often computers and servers, are made unavailable, often through request overloading.

2. Methodology

a. The Cyber Kill Chain

Our methodology for vulnerability discovery and exploitation, while it may vary from attack surface to attack surface, ultimately follows the seven steps of the cyber kill chain developed by Lockheed Martin[3]:

1. Reconnaissance: here is where we try to learn as much as we can about the system/function that we are hoping to attack. To clarify, the recon to which we are referring here is the research step for a specific attack surface, e.g. a web page login screen, and not the larger recon step with which we began the testing. The larger, initial recon step was to learn about the organization and its infrastructure as a whole. Here we hope to learn about a specific page or function. This step is driven by weaponized testing in which we use the page or function as intended to gain an understanding of its purpose and how it works.
2. Weaponization: this step is where we take what we learned about the system in the previous step, research historical/existing flaws in its deployment, and devise an input, payload, or action that will cause the system to behave in an unintended manner.
3. Delivery: here, we are simply providing the system with the "payload" that we generated in the previous step.
4. Exploitation: this step, if our weaponization tactic is successful, opens up the system to our malicious activity. This is where we can begin to confirm that the attack surface that we are targeting does in fact represent a genuine vulnerability. In turn, this might be where we confirm that the system is not vulnerable to our chosen payload.
5. Installation: following successful exploitation, this step is an extension of the previous one. We confirm that we have gained some level of unintended control over the target system.
6. Command and Control: by this point we have confirmed that a given vulnerability exists on this system, next we begin to gather data for determining this vulnerability's risk score by analyzing our access and determining which assets we have gained control over.

7. Actions on Objectives: finally, as an extension of our command and control, we take action on the assets that we have gained access to. This can include reading, changing, and deleting files within the system.

Upon completing these steps we have identified a valid vulnerability for reporting to our customer, Near-Earth Broadcasting Network (NBN), and gathered enough information about what actions are unlocked to the attacker to accurately score its risk to the organization. As mentioned previously, not all vulnerabilities in this report will show a copy-paste of this methodology, but much of the exploitation process in this test will reflect the actions defined in these steps.

b. Risk Scoring Method

To gain a clear understanding of the risk scores included within this report, we must describe how we identified the risk associated with a vulnerability. To do so, we employed a risk matrix provided by project-management.com [2]. This 5x5 matrix features 5 levels of severity/consequence on the x-axis and 5 levels of likelihood/ease of exploitation on the y-axis. Thus, in order to determine the risk score of a given vulnerability, we must determine, on a scale from 1 to 5, how likely it is that an attacker will discover and exploit this vulnerability and what are the consequences of this vulnerability being exploited. These two numbers will then be traced into the middle of the matrix to provide us with a risk score between 1 and 25 for the given vulnerability.

It is important to note that while this risk matrix system does an excellent job of providing a risk score based on the two most important details about a vulnerability, likelihood and consequence, it is not a perfect system. Specifically, certain vulnerabilities, while present and necessary to be patched, require certain prerequisite vulnerabilities to have existed. For example, a vulnerability in which a web server allows any user to open ports from the server's command terminal implies that the attacker has already gained access to said command terminal. So, while this creates some perhaps imperfect risk scoring results, for the purposes of our report we have chosen to proceed with the assumption that an attacker is granted any prerequisites to a given vulnerability. To return to our example, when scoring the users' ability to open ports, we score this vulnerability as though the attacker has already gained access to the server's command terminal.

c. Tools

We have included a list of software tools used during our testing:

- Nmap, network discovery tool [4]
- Hydra, network login cracker [5]
- Secure Shell Protocol (ssh), network protocol for secure remote login [6]
- File Transfer Protocol (ftp), communication protocol for file transfer between hosts and servers [7]

These software tools and protocols represent the toolkit used during our testing to identify and exploit the vulnerabilities disclosed in this report.

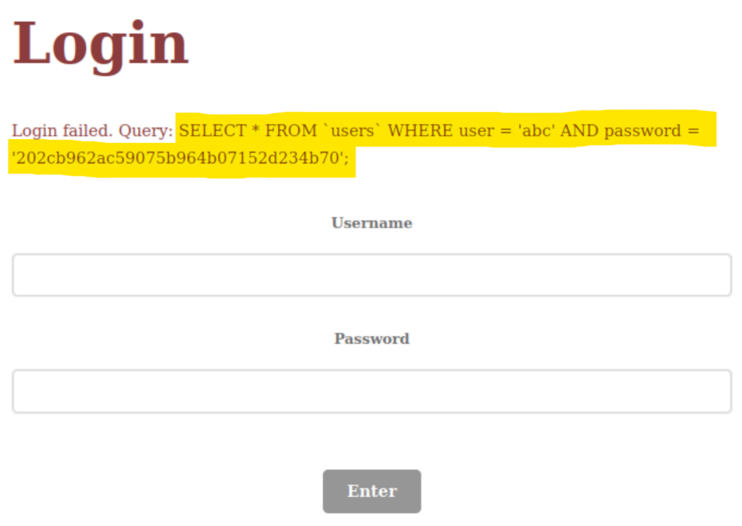
3. Findings

a. SQL - Failed Login Output

The first of our four vulnerabilities comes from the employee login page on Near-Earth Broadcasting Network's (NBN's) website. This page is found by first visiting the website's homepage, in which NBN has a news feed, lists details for consumers about their organization, and provides a form to subscribe to their experience. At the bottom of the link directory featured on the right of the page is a clickable link to the employee login screen. Here is where we identified our first vulnerability.

Traditionally, in a more secure implementation of a login screen, when invalid credentials are submitted, the web page will display a simple error message, e.g. "invalid credentials." However, in NBN's current implementation, along with text that states that the login was failed, the user is also shown the exact structured query language[2] (SQL) query that was used to test the credentials against the database entries. Examine this screenshot:

[2] Structured Query Language (SQL) is programming language for server-database interaction.

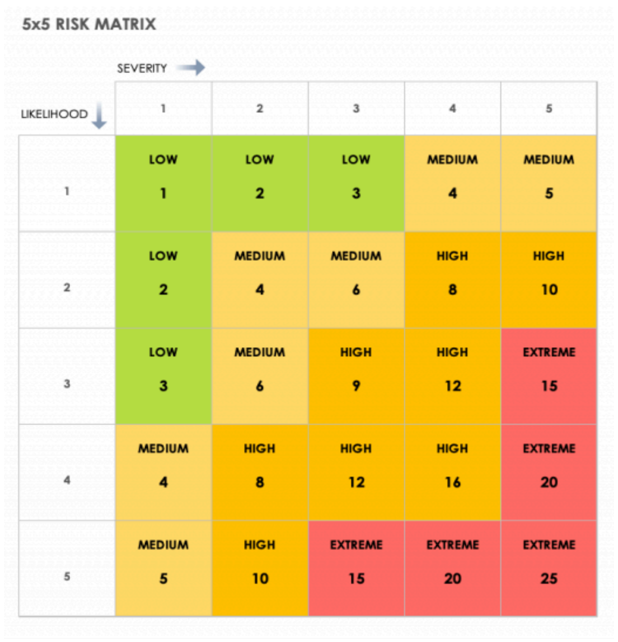

Login

Login failed. Query: SELECT * FROM `users` WHERE user = 'abc' AND password = '202cb962ac59075b964b07152d234b70';

Figure 1: Screenshot of failed login attempt at Employee Login page.

In this screenshot, for demonstration purposes, we submitted the invalid credentials: username=abc, password=123. In addition to printing the error "Login failed," the user is also shown the database query (highlighted in *Figure 1*): "SELECT * FROM `users` WHERE user = 'abc' AND password = '202cb962ac59075b964b07152d234b70';" This provides potential hackers with useful insight into how the database is queried. The attacker can see that the database behind the server is using the SQL programming language for queries, which opens the page up to potential SQL injection attacks[8]. Because we did not perform the SQL injection attack itself, we will not be explaining in detail how this attack is performed, instead we just wish to summarize how it may be used by attackers. By generating inputs that take advantage of how SQL queries are formatted, attackers can, in many cases, print to their screen the contents of the database without having to have used valid credentials. This can include usernames, passwords, and whatever other information about users is stored on the database. Thus, this vulnerability has the potential to leak user information and valid credentials to attackers.

To score this vulnerability, we examined the previously mentioned risk matrix in *Figure 2*:



5x5 RISK MATRIX

LIKELIHOOD ↓ / SEVERITY →	1	2	3	4	5
1	LOW 1	LOW 2	LOW 3	MEDIUM 4	MEDIUM 5
2	LOW 2	MEDIUM 4	MEDIUM 6	HIGH 8	HIGH 10
3	LOW 3	MEDIUM 6	HIGH 9	HIGH 12	EXTREME 15
4	MEDIUM 4	HIGH 8	HIGH 12	HIGH 16	EXTREME 20
5	MEDIUM 5	HIGH 10	EXTREME 15	EXTREME 20	EXTREME 25

Figure 2: 5x5 Risk Matrix by project-management.com

We determined that given how easy it was to view the failed SQL query, by simply submitting invalid credentials, that the likelihood that an attacker would see this output is a 5 (on the y-axis). As for severity, we scored this vulnerability as a 2. The consequence is only a 2 because the specific vulnerability that we are referencing, print the query used, is only the beginning of a much greater exploit. Just knowing what query language is used does not guarantee that an SQL injection attack will be successful. Thus, with a likelihood of 5 and a consequence of 2, our final risk score for this vulnerability is 10 (high).

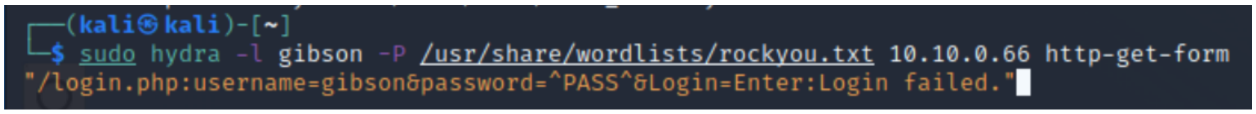
Because we did not perform the SQL injection itself, we labeled this vulnerability as an oversharing of error details. Thus, the fix that we recommend:

- Remove the database query as part of the output message,
- Only print that the login has failed, and
- Optionally, include a slightly more specific reason as to why it failed (e.g. “invalid username and/or password”)

These changes will ensure that an attacker is not gifted any insight into the database on which the login credentials are stored. The attacker will not be learning anything from submitting invalid credentials.

b. Hydra - Employee Password Guessing

The second vulnerability to report is the lack of a policy enforcing complex passwords for employee login credentials. In order to brute-force[3] "guess" a valid user's credentials, we employed Hydra, a network login cracker available in the Kali distribution of Linux. In short, returning to the login page from the previous vulnerability, our configuration of Hydra repeatedly attempted credentials until a match was found and a valid login was achieved. In order to do this, Hydra requires several fields to be set to initiate the attack. The final hydra command entered in our Linux command terminal can be seen in the screenshot below (*Figure 3*):



```
┌──(kali㉿kali)-[~]
└─$ sudo hydra -l gibson -P /usr/share/wordlists/rockyou.txt 10.10.0.66 http-get-form
"/login.php:username=gibson&password=^PASS^&Login=Enter:Login failed."
```

Figure 3: Hydra command to brute force user: gibson's password.

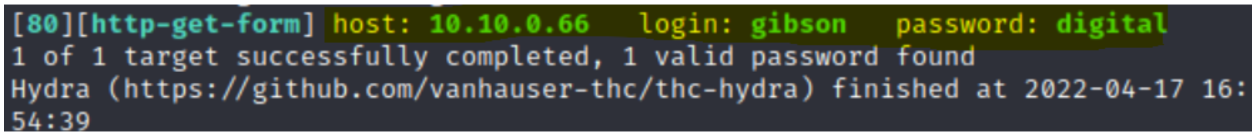
Below is a list of the required fields, what we entered, and how we chose what to enter:

- "sudo hydra" - command to run the hydra cracker with root privileges from our linux system
- " -l gibson" - specify the username to use in combination with password guesses. We noted that our contact, Bill Gibson, had the company email *gibson@corp.nbn*. We then correctly inferred that "gibson" was his username for the employee login page.
- "-P /usr/share/wordlists/rockyou.txt" - the path to a text file, rockyou.txt, a list of "common" passwords from the hack of company RockYou in 2009 [9]. This wordlist will be used to feed Hydra passwords to guess in combination with the username gibson.
- "10.10.0.66" - the IP address of the server's public facing interface. This is where the server can be reached by the public.
- "http-get-form" - the hyper text transfer protocol (http) method used behind the scenes when a user submits credentials for login. The HTTP GET method was chosen specifically after analyzing the network tab of Firefox's developer tools

[3] Brute-force password guessing is when an attacker tries many passwords taken from a wordlist until the correct one is found.

after credentials (valid or not) are submitted. This is the method that the web page employs to submit a login attempt.

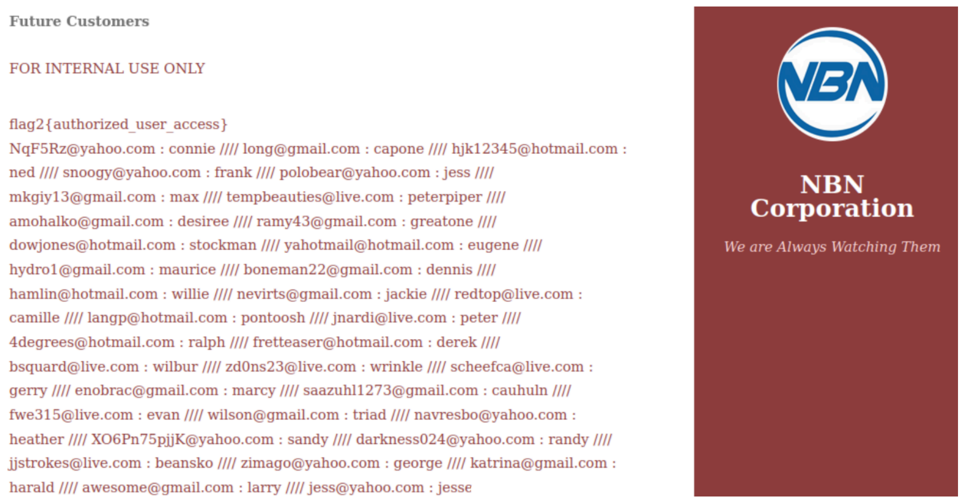
- "/login.php" - this specifies the path within the web server to the login page itself. It can be found by viewing the url bar after navigating to the login page: http://10.10.0.66/login.php
- "username=gibson&password=^PASS^&Login=Enter" - here we specify how hydra is to behave in order to mimic authentic activity on the web page. Hydra will enter gibson into the username field, a value from the wordlist into the password field, and "click" the enter button to submit the credentials.
- "Login failed." - this string provides Hydra with a way to know whether or not the credentials were valid and a successful login was achieved. Because a failed login will print "Login failed." to the page, Hydra will search the page for this string after every submission. If it is unable to find it, it knows that a login was successful, see *Figure 4*.



```
[80][http-get-form] host: 10.10.0.66   login: gibson   password: digital
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-04-17 16:
54:39
```

Figure 4: Hydra reporting a successful username and password combination has been found.

This Hydra command provided us with the following credentials: username=gibson, password=digital. From there, we were able to return to the employee login page in our browser and login as Bill Gibson. This gave us access to the "flag2{authorized_user_access}" file as well as a list of "future users" and their emails labeled "for internal use only" see *Figure 5*.



```
Future Customers

FOR INTERNAL USE ONLY

flag2{authorized_user_access}
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
```

Figure 5: output of successful employee login with valid user credentials.

This attack took Hydra less than a minute to find valid user credentials given only that an attacker knows a username. Further, to that point, usernames can often be easy enough to guess given publicly available employee records.

Next, we return to the risk matrix to report our risk score for this vulnerability (see *Figure 2*). We determined that while this vulnerability requires some knowledge of the HTTP protocol and the hydra software, if the attacker has such knowledge, it is a quick and easy exploit. Thus, we scored this vulnerability with a likelihood of 3. For consequence, because a successful login leaks the flag2 file and customer information, we scored it a 5. Customer information leakage can negatively impact reputation, and subsequently cost a lot financially to recover that reputation. With a likelihood of 3 and a consequence of 5, the overall risk score for this vulnerability is a 15 (extreme).

Our immediate recommendations for patching and hardening to discourage from this attack are as follows:

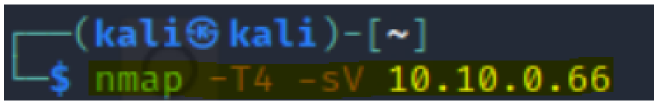
- Stronger employee password requirements, this should likely include integrating a password complexity plugin like PassRequirments for jQuery [1]
- Multi factor authentication, passwords are simply something you know, we recommend adding a layer of something you are (e.g. biometrics), or more financially viable, something you have (e.g. a confirmation code sent via email or text)
- Account locking after multiple failed attempts

Each of these three options are an improvement in and of themselves and will improve NBN's cyber security stature, but we highly recommend the inclusion of all three. This will discourage password guessing through brute forcing methods and help ensure that the only user logging into an account is the account owner themselves.

c. ssh - remote login and file browsing

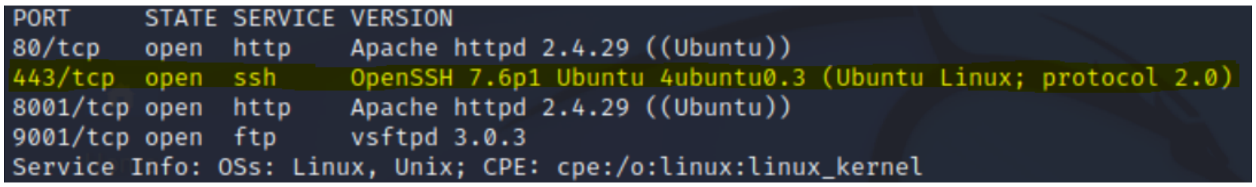
The third vulnerability to report is found by utilizing the credentials found with Hydra in the previous vulnerability. Let us recap, in our previous vulnerability, we utilized the Hydra network login cracker to discover Bill Gibson's credentials: username=gibson, password=digital. Note that these credentials serve as login material for the server. As such, our target for this next vulnerability is the server, in this case at 10.10.0.66. Here, we create a remote, secure terminal connection using the Secure Shell Protocol (ssh) [6]. To summarize, ssh allows us to remotely login with gibson's credentials, send commands, and ultimately open a command terminal on the server that runs on our screen. Note that valid credentials are required for this step, that said, we discussed earlier that for the purpose of this report, we assume that an attacker can discover these credentials.

To perform an ssh login with our newly found credentials, we first needed to identify which port on the server is open to ssh protocol connections. To do so, we used the nmap network scanner included in the Kali distribution of Linux (among other distributions). First, see our nmap command used in *Figure 5*, and the the output of the command in *Figure 6*.



```
┌──(kali㉿kali)-[~]
└─$ nmap -T4 -sV 10.10.0.66
```

Figure 5: nmap command to scan the server's ports.



```
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))
9001/tcp open  ftp     vsftpd 3.0.3
Service Info: OSs: Linux, Unix; CPE: cpe:/o:linux:linux_kernel
```

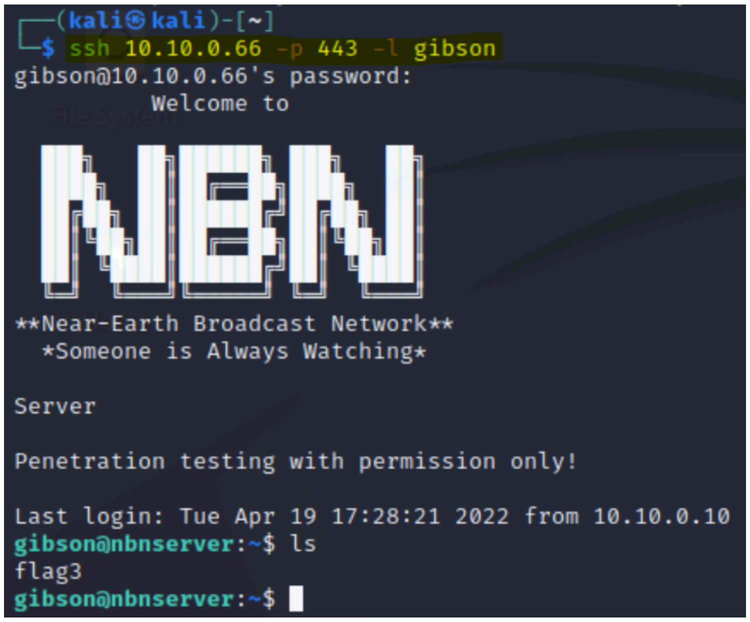
Figure 6: output of nmap command, port 443 open to ssh connections.

To provide a quick summary of the nmap command, here is a list of the flags used and their purpose:

- "-T4" - a 4 out of 5 possible speed options, 5 being the fastest.
- "-sV" - this includes the services running on the open port within our output. See the "service" tab in the nmap output in *Figure 6*

This command provided us with our output as seen in *Figure 6*. The highlighted entry in the nmap output shows that, by using nmap, we discovered the port 443 on the server was open to ssh connections.

Now that we have identified which port to use, we were ready to launch our ssh login request. We entered the following highlighted command in our terminal (*Figure 7*):



```
┌──(kali㉿kali)-[~]
└─$ ssh 10.10.0.66 -p 443 -l gibson
gibson@10.10.0.66's password:
          Welcome to

**Near-Earth Broadcast Network**
  *Someone is Always Watching*

Server

Penetration testing with permission only!

Last login: Tue Apr 19 17:28:21 2022 from 10.10.0.10
gibson@nbnserver:~$ ls
flag3
gibson@nbnserver:~$ 
```

Figure 7: ssh command and successful connection establishment.

Let us now clearly define each aspect of our command to establish the remote connection:

- "ssh"- use the ssh protocol to initiate a remote connection
- "10.10.0.66" - specify the IP address of the host that we wish to connect to, in this case the server at 10.10.0.66
- "-p 443" - the port open for ssh as identified using nmap (see *Figures 5* and *6*)
- "-l gibson" - the username identified in a previous vulnerability

Allowing an employee to login remotely is not in and of itself a vulnerability, but as an attacker we now have all of the privileges of the gibson user. To demonstrate the dangers of this vulnerability: we viewed the passwd file located in the /etc/ directory and the flag3 file located in the /gibson/ directory. The passwd file keeps a record of all of the registered users on the system. The flag3 file is one of the previously mentioned ctf-style flags representative of sensitive company information. These files are just two examples of the sensitive data that was available to us as attackers by exploiting this vulnerability.

Next, the risk score for this vulnerability includes the assumption that valid user credentials, like the ones found in our previous vulnerability, have already been acquired by the attackers. We scored the likelihood of exploitation to be a 5. Once credentials have been recovered, the only other requirement on the attacker's end is a baseline understanding of nmap to determine if and which port is open to ssh connections. We figure that even the most inexperienced attacker knows their way around nmap. As such, it becomes as simple as entering the ssh command to launch the remote login. For consequence of exploitation, this vulnerability scored a 3. While we were able to print the contents of the passwd file, none of the passwords are stored in plaintext. So, while attackers viewing this file is not proper security practice, it also does not immediately leak every users' credentials. That said, access to the company secrets within the flag3 file, including the "flag3{brilliantly_lit_boulevard}" string has the potential to severely damage NBN's reputation and financial stature. Thus, the final risk score for the ssh remote login and file access vulnerability is 15 (extreme). This is the highest risk score so far and we highly recommend immediate actions to address this vulnerability.

Our recommendations for actions to address this vulnerability and improve the organization's data privacy are as follows:

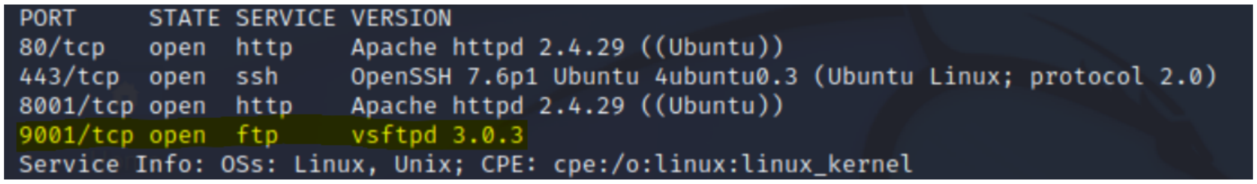
- The appropriate department within the organization should be regularly reviewing logs to detect any unordinary and/or malicious activity performed through user accounts
- The current logging system detects when a remote ssh login is performed and writes it to the "auth.log" file located in the /var/log/ directory, but we recommend that such actions create an alert on the system that notifies the appropriate personel to perform an immediate review

These two steps will help prevent ssh logins by malicious users by keeping the organization privy to users' activities from outside the network. The organization will be made aware of any remote sessions that are created and can cross reference them to the expected behavior of its employees.

d. ftp - remote file download

Our final vulnerability to report is the File Transfer Protocol (ftp) file download. This vulnerability serves as something of an extension of the previous vulnerability involving an ssh remote connection. Once again, the target of this exploit is the server, once again running with the IP address 10.10.0.66. This exploit is taking advantage of an open port on the server made available for the ftp protocol. The ftp protocol is used for file transfers under the client-server model. In this case, our attacker host is the client and NBN's server is the server. An ftp connection to NBN's server requires a login, much like ssh did, so we reused the gibson/digital credentials uncovered earlier.

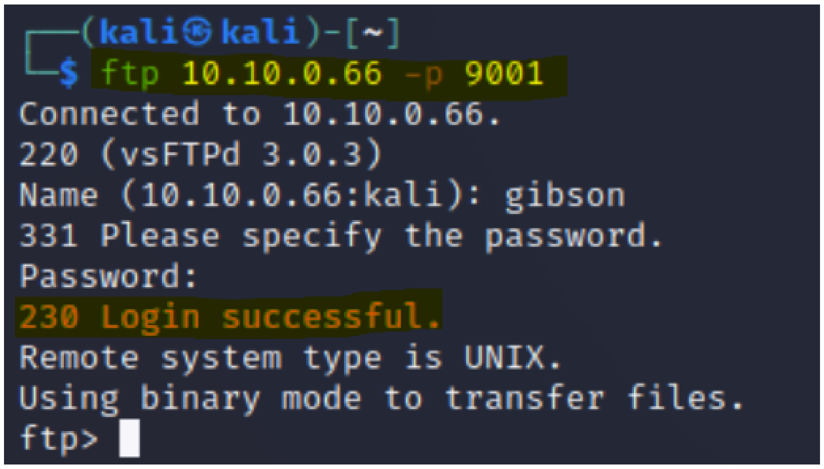
Like ssh, our exploitation of this vulnerability begins with a nmap scan to identify if the server contains an open port for ftp. We deploy the same nmap scan as previously used in *Figure 5*, but we revisit our output to locate an open ftp port (see *Figure 8*):



```
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))
9001/tcp open  ftp     vsftpd 3.0.3
Service Info: OSs: Linux, Unix; CPE: cpe:/o:linux:linux_kernel
```

Figure 8: nmap scan output with open ftp port highlighted.

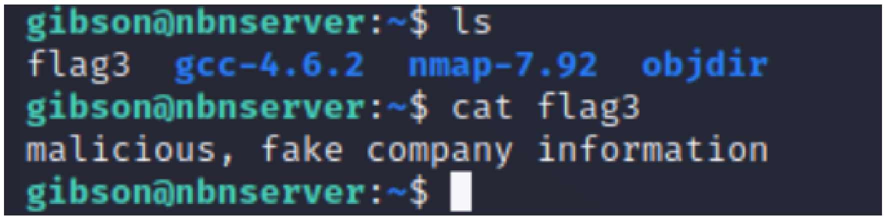
Our scan revealed that port 9001 is open for ftp connections, specifically running vsftpd (highlighted in *Figure 8*), the default ftp server for many Linux distributions. This port represents the last detail needed to launch our ftp command and connect to the server for file transfer. The command entered on our terminal is included in *Figure 9*:



```
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.0.66 -p 9001
Connected to 10.10.0.66.
220 (vsFTPd 3.0.3)
Name (10.10.0.66:kali): gibson
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Figure 9: ftp command with successful login confirmation

The command, highlighted in yellow, includes the target server IP address, NBN's server at 10.10.0.66, and the target port 9001 identified with the nmap scan. The figure shows the successful ftp login using the gibson credentials highlighted in orange. Once connected to the system, we as attackers were able to download any files that the gibson user has access to, this included the sensitive flag3 file discussed previously. Further, we are able to upload any files from our attacker machine to the server. To demonstrate the dangers of both malicious downloading and uploading, we stole the flag3 file (downloaded to our machine), and replaced it with a malicious flag3 file that we created and uploaded from our machine. We replaced the original flag3 file with the malicious one shown in *Figure 10*:



```
gibson@nbnserver:~$ ls
flag3  gcc-4.6.2  nmap-7.92  objdir
gibson@nbnserver:~$ cat flag3
malicious, fake company information
gibson@nbnserver:~$ 
```

Figure 10: The "cat" command prints the contents of a file; The NBN server now contains a malicious flag3 file.

Now what if this file contained truly sensitive info like usernames and passwords or customer credit card information? On top of stealing and replacing the flag3 file, we also downloaded the passwd file described earlier. Thus, we have demonstrated the dangers of such an ftp vulnerability. Unchecked file transferring has the potential to leak sensitive company data, and call into question the integrity of data residing on the server.

To score this vulnerability, we considered how easy it is to launch an nmap scan and to connect to a server via ftp. That said, we noted that the gibson user did not have root privileges, so luckily some sensitive files remained unattainable via his credentials. As such, the likelihood score chosen was 4. Next, because of the dangers of such a vulnerability, as demonstrated in our testing, we scored the consequence of exploitation as a 4. This brings our total risk score for this vulnerability to a 16 (high). This high score puts this ftp - unchecked file transfer vulnerability near the previously reported ssh vulnerability as highly dangerous and in need of immediate patching.

To patch such a vulnerability, we recommend three essentials steps:

- Close monitoring and regular review of the ftp connections that are already being logged within the vsftpd.log file located in the /var/log/ directory. Regular cross-referencing of these file transfers with expected user behavior will alert the organization to potentially malicious activity relating to ftp connections.
- Storing a hash[4] of the file for integrity checks. This allows users to perform a check to see if the file has been tampered with outside of official changes.
- Similar to our recommendation for ssh, ftp file transfers should trigger alerts that notify the appropriate personnel for immediate review.

The combination of these three policy changes will make it much more difficult for attackers to abuse the open ftp port without NBN being made aware of the activity. This will not completely seal NBN off from this type of exploitation, but it will reduce the risk by better equipping the organization to find and terminate these malicious activities. This concludes the main body of our report.

4. Conclusion

While we have reached the end of this report, there are several points that we wish to reiterate. Apex Security Consulting (ASC) was contracted by Near-Earth Broadcasting Network (NBN), in April of 2022, to perform a black-box style penetration test on their systems. For the purpose of this test, as not to perform dangerous attacks on NBN's live network, ASC was provided with two virtual machine images. The two images, a server system hosting NBN's website and a host system representative of the

[4] Hashing is the process of passing some data through what is called a hash function that produces a fixed size output representation of the original data that does not reveal anything about the original data. Hashing functions are one-way functions, meaning it is difficult to compute the input given the output.

internal staff computers, were created to reflect NBN's live infrastructure while being run in an enclosed environment and therefore safe to attack.

NBN's goal with this penetration test was to identify vulnerabilities within their network stack, system configurations, and infrastructure that expose them to exploitation by external threat actors. As external threat actors, the team at ASC was not provided with any additional insights, beyond those available to the general public, about NBN's systems. Our goals included:

- Identifying weaknesses in the public facing server that enable attacker access to the internal network,
- Locate weaknesses in devices or services within the network that permit further malicious activities to be performed by attacker, and
- Report these vulnerabilities in a detailed manner including how they were discovered, what level of risk do they pose to the organization, and what options exist to address these weaknesses.

All of these goals were outlined for ASC with the intention of improving NBN's cyber security posture after implementing the patches recommended in this report.

ASC's test identified four vulnerabilities that introduce varying degrees of risk to the company. Firstly, an overindulgence of query language details being printed upon a failed employee login attempt. This excessive sharing of details scored a 10 (high) within our risk scoring matrix. Next, the insufficient password complexity for employee accounts allowed us as attackers to enumerate our NBN contact's login credentials. This insufficient password strength scored a 15 (extreme) on our risk matrix. These login credentials then opened up NBN to two more vulnerabilities, a secure shell protocol login and directory traversal, and a file transfer protocol for file transportation. Returning to our risk matrix, these two vulnerabilities scored a 15 (extreme) and 16 (extreme), respectively.

Taking an average of these scores determined that NBN's final risk score, as determined by this penetration test, is a 14 (high). We encourage NBN to take advantage of the information provided in this report and take action to improve their security posture. In its current state, even a novice hacker has options to exploit certain vulnerabilities in NBN's infrastructure.

While we defer NBN to the findings section of this report to review important actions for addressing the vulnerabilities discussed here, we do wish to reexamine some important options for immediate fixes. NBN must disable the SQL query output upon a failed attempt within the employee login page of their website. At no point in the login process should users be provided with any details about the underlying database, including what query language is in-use. The response to a failed login should be simplified to a simple “Login failed” message. An optional addition within that message is a short explanation as to why the attempt failed, e.g. incorrect password.

The next essential fix to be deployed immediately is a system to require at least a certain level of complexity for employee account passwords. We recommend a plugin for the existing software such as PassRequirement[1] for jQuery. This plugin contains a list of password complexity requirements to be satisfied at the time of password creation. These requirements include details like password length, special character inclusion, and variety between lower-case letters, upper-case letters, and numbers.

By implementing these fixes and taking the time to review this report, we expect the Near-Earth Broadcasting network can improve their cyber security stature and deliver on customer expectations of data privacy.

3. Appendix A: Tools

1. nmap : https://nmap.org/
2. hydra: https://www.kali.org/tools/hydra/
3. ssh: https://man.openbsd.org/ssh.1
4. ftp: https://en.wikipedia.org/wiki/File_Transfer_Protocol

4. Appendix B: References

1. 09arnold. (2018, February 12). *09arnold/passrequirements: A jQuery plugin that lists a set of requirements for an input field and crosses them out as the input matches each "requirement".* GitHub. Retrieved April 16, 2022, from https://github.com/09arnold/PassRequirements
2. Walker, B. (2022, February 22). *Risk Assessment Matrix 2022*. Project Management. Retrieved April 18, 2022, from https://project-management.com/risk-assessment-matrix/
3. *Cyber kill chain®*. Lockheed Martin. (2022, April 11). Retrieved April 18, 2022, from https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html
4. Nmap. (2020, October 3). Retrieved April 10, 2022, from https://nmap.org/
5. *Hydra: Kali linux tools*. Kali Linux. (2022, February 10). Retrieved April 11, 2022, from https://www.kali.org/tools/hydra/
6. *OpenBSD Manual Page Server - ssh*. ssh(1) - OpenBSD manual pages. (n.d.). Retrieved April 12, 2022, from https://man.openbsd.org/ssh.1
7. Wikimedia Foundation. (2022, April 17). *File transfer protocol*. Wikipedia. Retrieved April 12, 2022, from https://en.wikipedia.org/wiki/File_Transfer_Protocol
8. *SQL injection*. PortSwigger. (n.d.). Retrieved April 26, 2022, from https://portswigger.net/web-security/sql-injection
9. Burns, W. J. (2019, January 13). *Common password list (rockyou.txt)*. Kaggle. Retrieved April 11, 2022, from https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt

10. Harley. (2021, July 29). *How to Brute Force websites & online forms using hydra*. Infinite Logins. Retrieved April 11, 2022, from https://infinitelogins.com/2020/02/22/how-to-brute-force-websites-using-hydra/

11. Wright, G. (2021, July 21). *What are ports in computing and how do they work?* SearchNetworking. Retrieved April 17, 2022, from https://www.techtarget.com/searchnetworking/definition/port

12. *Network protocol definition: Computer protocol: Computer networks: Comptia*. Default. (n.d.). Retrieved April 18, 2022, from https://www.comptia.org/content/guides/what-is-a-network-protocol

13. *Penetration test report - offensive security*. Offensive Security. (2013, August 10). Retrieved April 17, 2022, from https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf

14. *Root account*. IBM. (n.d.). Retrieved April 18, 2022, from https://www.ibm.com/docs/en/aix/7.2?topic=passwords-root-account

15. *Client-server model*. GeeksforGeeks. (2019, November 15). Retrieved April 18, 2022, from https://www.geeksforgeeks.org/client-server-model/

5. Appendix C: Sensitive Data Enumerated

1. flag2: flag2{authorized_user_access}
2. flag3 file
3. flag3: flag3{brilliantly_lit_boulevard}
4. Bill Gibson employee credentials: username - gibson; password - digital
5. passwd file

6. Appendix D: Glossary of Terms

1. Exploit: A threat event that weaponizes code or an application, to take advantage of a weakness for the purpose of having an intended effect on a target that would otherwise be impossible, unintended by the target owner, or unauthorized.
2. Vulnerability: A weakness in a business process, configuration, operating system, or application that can be used to create unintended and undesired scenarios.

3. Asset: Something that holds value, whether it’s a system or data, that a threat may be trying to access or compromise.
4. Black Box Testing (Red Team): Not provided any details on the target, system, network, application, or asset.
5. Payload: code to execute an exploit.
6. Port: A port in networking is a software-defined number associated with a network protocol that receives or transmits communication for a specific service [11].
7. Protocol: an established set of rules that determine how data is transmitted between different devices in the same network [12].
8. Root privileges: virtually unlimited access to all programs, files, and resources on a system [14].
9. Client-server model: a distributed application structure that partitions task or workload between the providers of a resource or service, called servers, and service requesters called clients [15].