Fleming College

Course	COMP 357 – Advanced Pentesting
Lab Assignment	Purple Team Guide
Instructor	Abeeeeee
Section	L02
Student Name	Manuel Manrique Lopez & Ricardo Rubin
Submission Date	December 07th, 2025

Purple Team Guide.

This scenario takes place after a successful browser exploitation attack using BeEF. The attacker initially established remote control over the victim's browser using a malicious webpage (phishing technique).

Now, the goal of the Purple Team is to deploy security controls, harden the browser, and verify that the original attack vectors are no longer effective.

These defenses will demonstrate the following:

- Blocking external malicious scripts
- Preventing BeEF remote command execution
- Reducing attack surface inside the browser
- Increasing user awareness to avoid future phishing attacks

This purple team activity reflects real-world browser security requirements for banking, enterprise access, and confidential applications.

Mitigation Goals

- Prevent malicious JavaScript execution (BeEF payload)
- Deny WebSocket/WebRTC communications used by attackers
- Restrict unnecessary browser components and extensions
- Encourage user phishing detection and link validation skills

If you followed the instructions of my Attack Report and GitHub repository, your environment should already contain:

- A Windows browser previously exploited
- A BeEF server running on Kali Linux
- A malicious webpage hosted via HTTP server

Now we will **block** that attack path by applying secure configurations.

Browser Hardening (Chrome Flags)

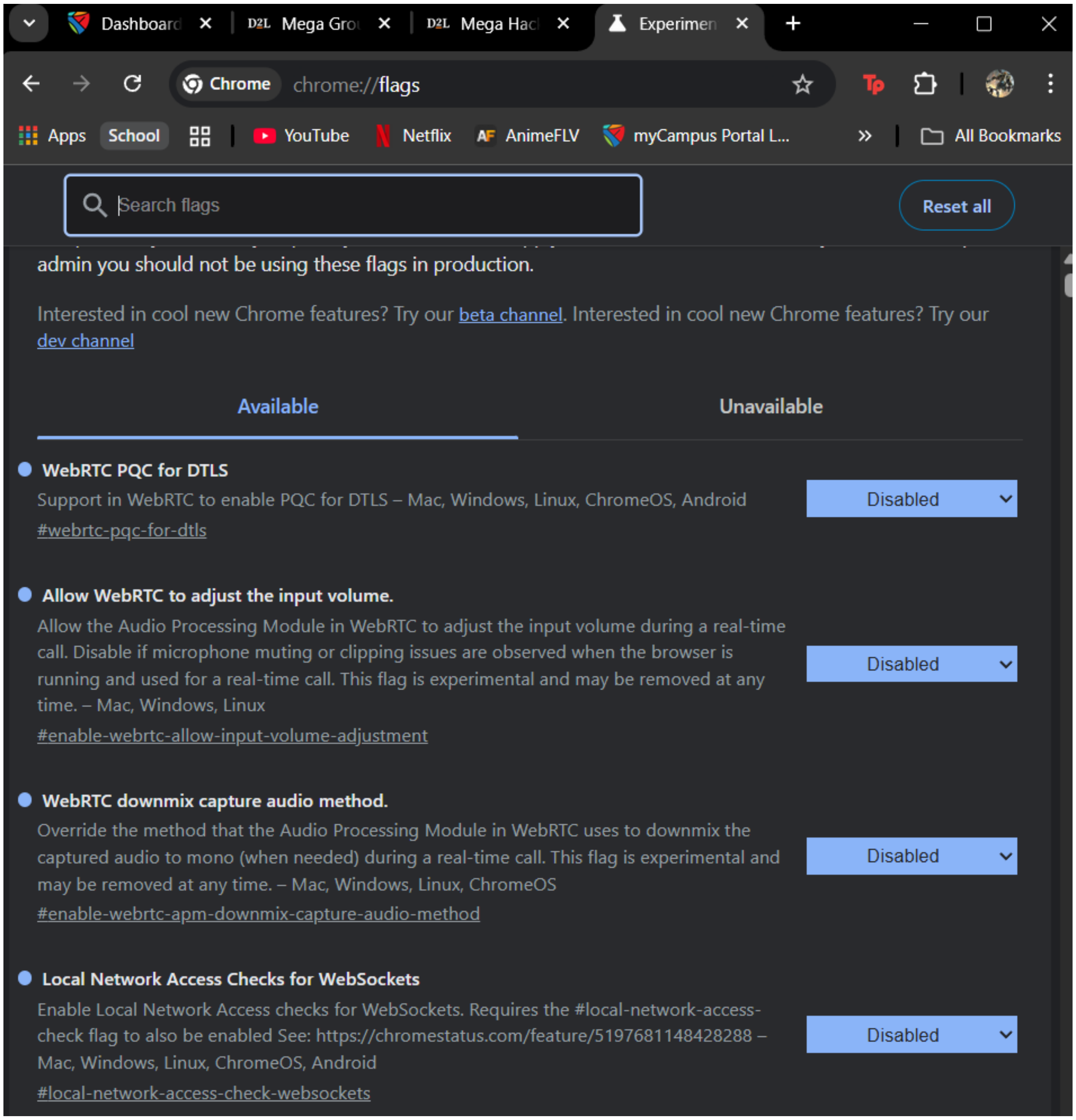
The victim browser must disable technologies commonly abused for post-exploitation actions such as:

- WebRTC
- WebSockets network access

Navigate to: ***chrome://flags/***

Then set these flags to Disabled:

- WebRTC PQC for DTLS
- Allow WebRTC to adjust input volume
- Local Network Access Checks for WebSockets



admin you should not be using these flags in production.

Interested in cool new Chrome features? Try our beta channel. Interested in cool new Chrome features? Try our dev channel

Available | Unavailable

WebRTC PQC for DTLS — Disabled
Support in WebRTC to enable PQC for DTLS – Mac, Windows, Linux, ChromeOS, Android
#webrtc-pqc-for-dtls

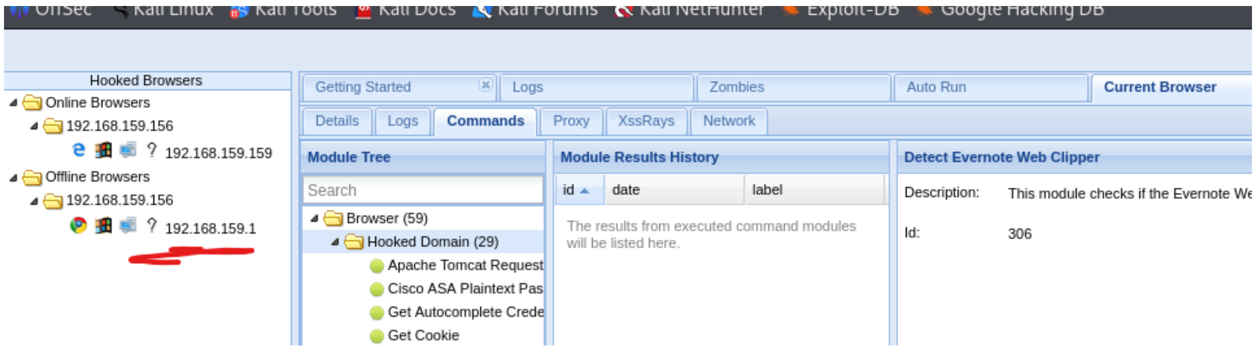
Allow WebRTC to adjust the input volume. — Disabled
Allow the Audio Processing Module in WebRTC to adjust the input volume during a real-time call. Disable if microphone muting or clipping issues are observed when the browser is running and used for a real-time call. This flag is experimental and may be removed at any time. – Mac, Windows, Linux
#enable-webrtc-allow-input-volume-adjustment

WebRTC downmix capture audio method. — Disabled
Override the method that the Audio Processing Module in WebRTC uses to downmix the captured audio to mono (when needed) during a real-time call. This flag is experimental and may be removed at any time. – Mac, Windows, Linux, ChromeOS
#enable-webrtc-apm-downmix-capture-audio-method

Local Network Access Checks for WebSockets — Disabled
Enable Local Network Access checks for WebSockets. Requires the #local-network-access-check flag to also be enabled See: https://chromestatus.com/feature/5197681148428288 – Mac, Windows, Linux, ChromeOS, Android
#local-network-access-check-websockets

Doing this will isolate the browser from remote command channels

- As soon as the page is refreshed, BeEF will lose connection and show the browser as Offline



Online Browsers
192.168.159.156
192.168.159.159
Offline Browsers
192.168.159.156
192.168.159.1

Browser (59)
Hooked Domain (29)
Apache Tomcat Request
Cisco ASA Plaintext Pas
Get Autocomplete Crede
Get Cookie

The results from executed command modules will be listed here.

Description:	This module checks if the Evernote We
Id:	306

- All modules executed by BeEF will start failing

This validates that the attacker can no longer control the compromised browser.

Strict Content Security Policy

Now we will block external JavaScript like BeEF's hook.js payload using Content Security Policy (CSP).

Edit your index.html and update the <head> section to:

<head>

<meta http-equiv="Content-Security-Policy" content="script-src 'self'; object-src 'none'; connect-src 'self';">

<title>Security Test Page</title>

</head>

This line ensures that:

- Only same-origin scripts are allowed
- The external BeEF hook is blocked entirely
- No connection can be established to the attack server

After applying CSP:

- Visit the malicious webpage again
- Open browser DevTools → Console

You will see an error message such as:

"Refused to load the script 'http://192.168.xxx.xxx:3000/hook.js' because it violates the document's Content Security Policy."

By implementing Content Security Policy and disabling WebRTC/WebSockets, we successfully mitigated the original exploitation path used by BeEF. The attacker lost the ability to maintain persistence, extract data, or perform internal reconnaissance. Enhancing user awareness of not falling on phishing emails and open unknown links is extremely important since making a human do a mistake is easy and probably the best way to inject a payload into someone’s server or even yourself.