Purple Team Mitigation Report – BeEF

Name: **Eric Jacob Daley**
Student ID: **10226039**
Course: **COMP 357**
Section: **01**
Due date: **12/?/25**
Instructor: **Abe Abernathy**

Browser Exploitation Framework (BeEF)

BeEF can hijack sessions, steal data, or otherwise wreak havoc by hooking browsers via JavaScript, often through phishing links or malicious sites.

This report presents practical strategies to prevent BeEF-style exploits. However, full disclosure: I'm relying on near-complete faith on two sources, so hopefully, this makes sense because I'm too busy (and/or too lazy) to test every step. Also, I stole Nick's formatting, and I haven't decided if I like it.

Mitigation Strategies

Keep Browsers and Plugins Updated

Outdated browsers and plugins are like an indoor basil plant left in a poorly ventilated room, only to be surprised when mold appears: unattended, small issues can quickly turn problematic.
Enable automatic updates and remove or update plugins like Flash, Java, and Adobe Reader, which are common targets for BeEF-style attacks.

Control JavaScript Execution

BeEF relies heavily on JavaScript, so controlling its execution drastically reduces risk.
Disable JavaScript globally or use extensions such as NoScript (Firefox) or ScriptSafe (Chrome) to selectively trust sites. This may break some website features, and marketing will likely complain, but it stops malicious hooks.

Implement Content Security Policy (CSP)

A Content Security Policy helps control which scripts and resources can run in your browser. Think of it as a strict kitchen manager: only trusted ingredients are allowed, and everything else is rejected. Key concepts include:

- Restrict script execution to approved sources via nonce or hash-based directives.
- Disable unsafe inline scripts and risky functions like eval().
- Block unnecessary embeds using object-src 'none'.
- Prevent clickjacking and enforce HTTPS where possible.
- Test first with a report-only policy to catch violations without breaking functionality.

The goal is to prevent malicious code from executing while maintaining necessary functionality, essentially, making sure your basil doesn't get overwhelmed by poor-quality additions.

Phishing Awareness and Link Hygiene

Since BeEF often enters through phishing, users should treat unexpected links like the menu at a cheap restaurant: sure, it claims to use basil, but it's probably poor-quality dried basil. Verify URLs, avoid pop-ups, and rely on browser anti-phishing tools.

Endpoint Protection and Firewalls

Maintain up-to-date antivirus software to detect malicious scripts or network activity. Configure firewalls to monitor and restrict outgoing connections to untrusted destinations, limiting what an attacker could do if a browser is compromised.

Limit Browser Permissions

Disable camera, microphone, geolocation, and notification permissions unless required. Be mindful of permissions granted to sites, as they can be exploited to gather data or launch attacks.

Conclusion

I probably could have implemented most of these fixes on a VM in the time it took me to force basil into every paragraph, but it's far too late for regrets. I've also decided I don't like Nick's formatting choices; center alignment has its moments, but bullet points are impossible to make look good, and honestly, who uses Aptos unironically? I did the quiz halfway through writing this, and now Rick Astley is living rent-free in my head, so I'll leave you with this:

Probably gonna giveeee up
Probably gonna let you down
Probably gonna let your browsers get hooked and exploited.

Sources

https://www.cyberly.org/en/how-do-you-protect-your-browser-from-beef-exploitation/index.html
https://developer.mozilla.org/en-US/docs/Web/Security/Practical_implementation_guides/CSP
https://genius.com/Rick-astley-never-gonna-give-you-up-lyrics
https://livetoplant.com/how-to-use-fans-to-strengthen-indoor-plant-stems