Gotta phish ém all !

Novel Attack Techniques via

Persistent Browser-in-the-Middle

Giacomo Lenzini

$ whoami

Giacomo Lenzini

@GiacoLenzo2109

Offensive Security Specialist @ EY Italy

- ➔ Red Team & Adversary Emulation / Simulation
- ➔ Penetration Testing
- ➔ Vulnerability Research

giacolenzo2109.github.io

giacomo-lenzini

GiacoLenzo2109

1.

The State of Phishing

Where It Breaks

1.1 The State of Phishing

The Adversary's Comfort Zone


Me, sending a standard
phishing email and get reported

1.2 The State of Phishing

The Main Critical Limitations

Limited MFA Bypass

Fails unless the victim is actively proxied in real time.

Post-Login Visibility

Zero visibility into the post-authentication session phase.

Operator interaction

No module dispatch, no live social engineering, no guided payload delivery.

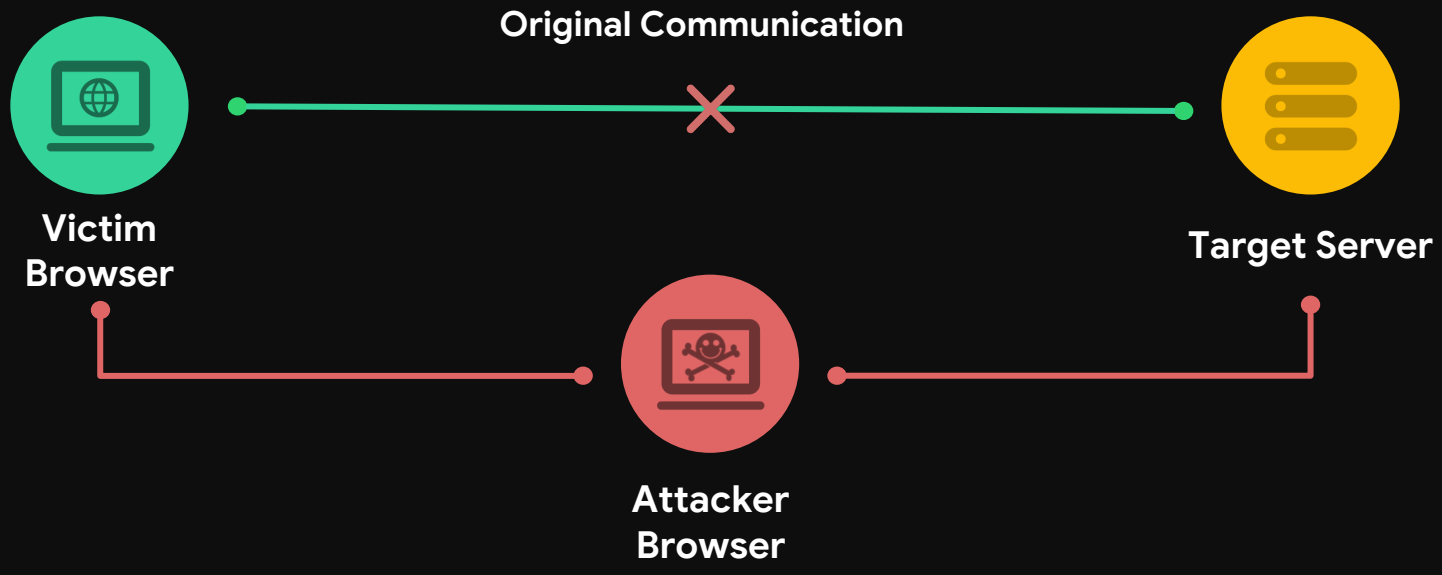
2.

Browser-in-the-Middle

The Paradigm Shift

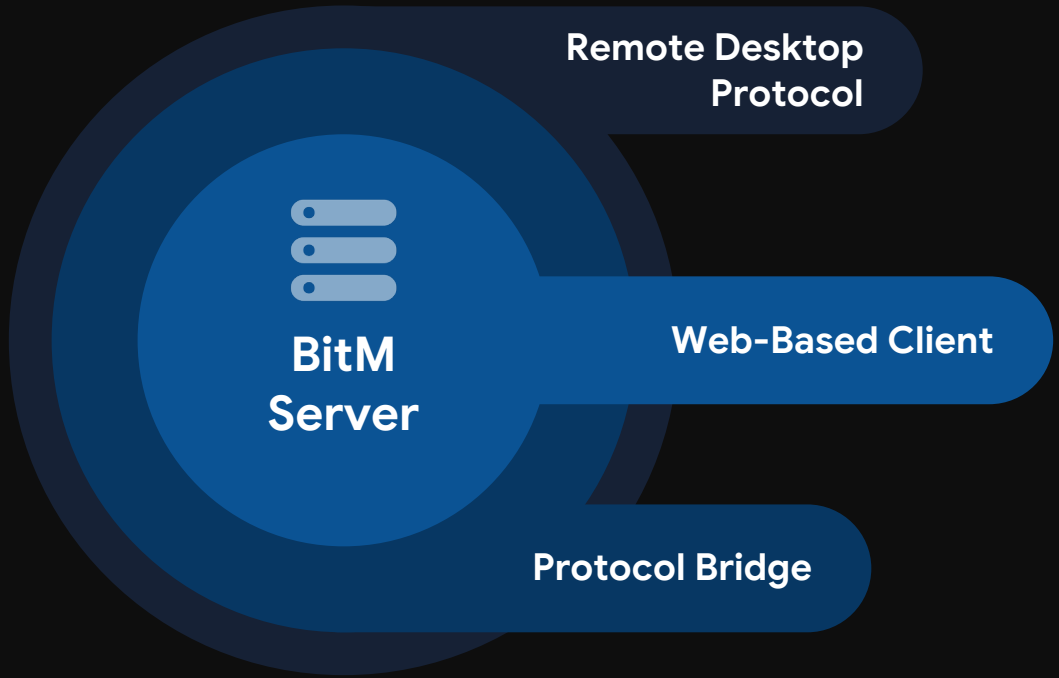
2.1 Browser-in-the-Middle

Attack Scheme that defeats MFA



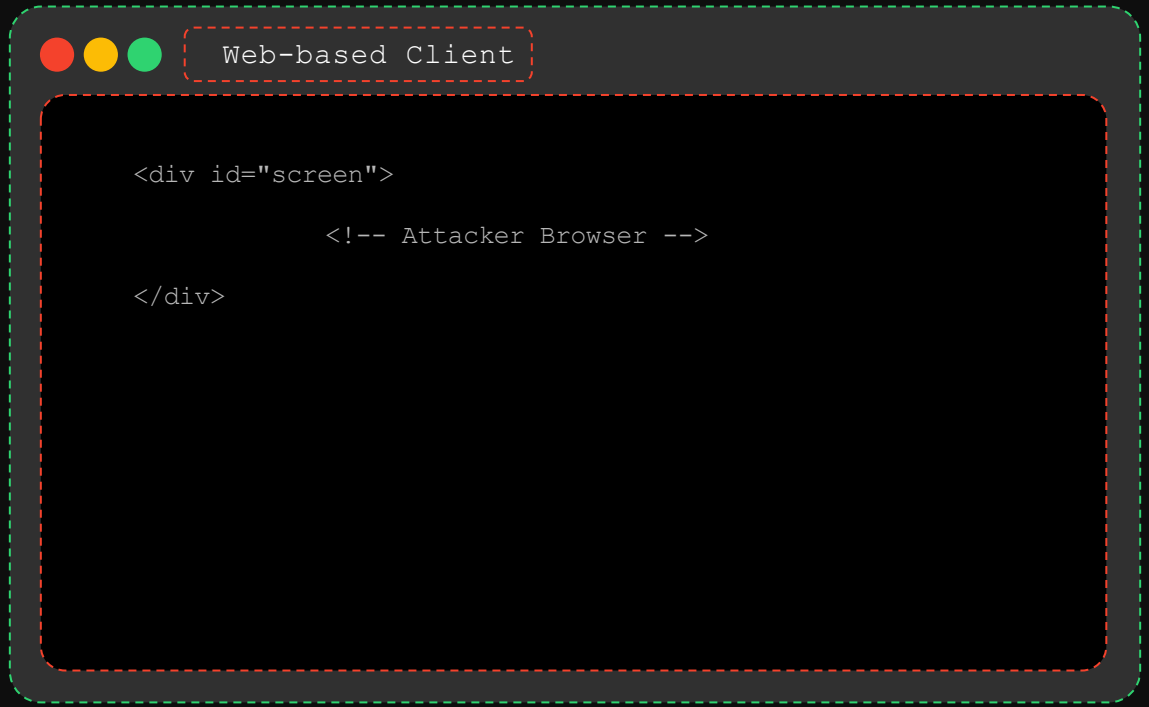
2.2 Browser-in-the-Middle

Attacker Infrastructure Component Breakdown



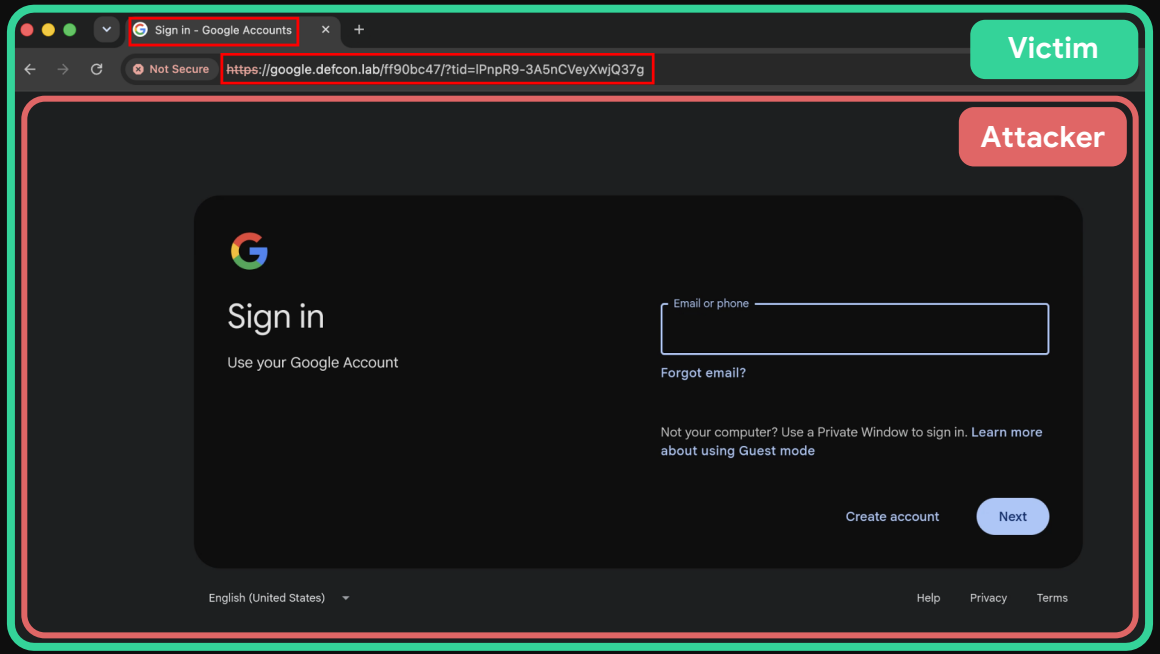
2.3 Browser-in-the-Middle

Victim Browser



2.3 Browser-in-the-Middle

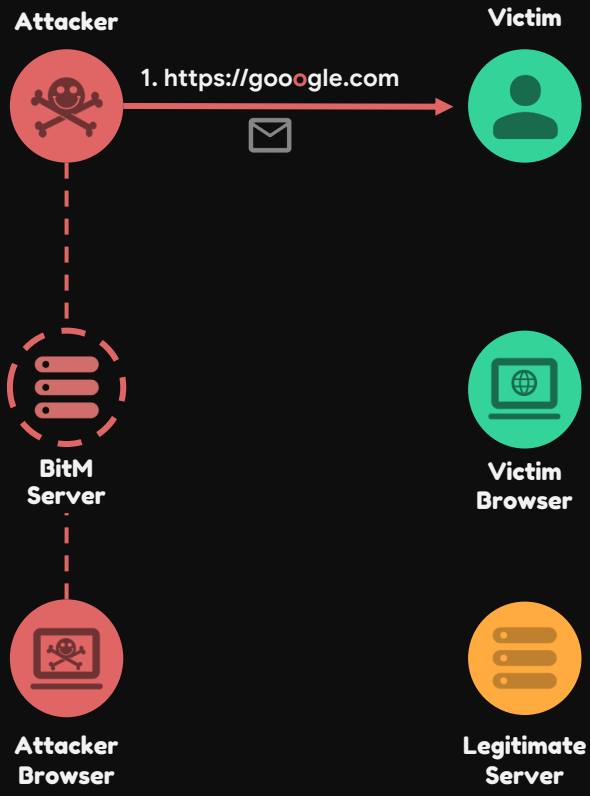
Victim Browser Example



2.4 Browser-in-the-Middle

Attack Flow

1. Phishing Link Delivery

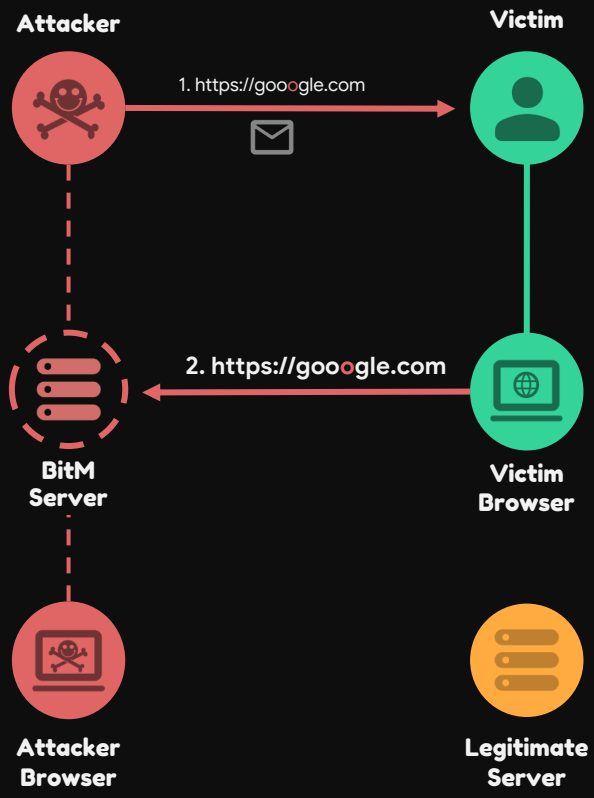


2.4 Browser-in-the-Middle

Attack Flow

1. Phishing Link Delivery

2. **Remote Browser Connection:** Instead of loading a fake cloned site, the victim instantly connect to a live, interactive stream of the attacker's remote browser.



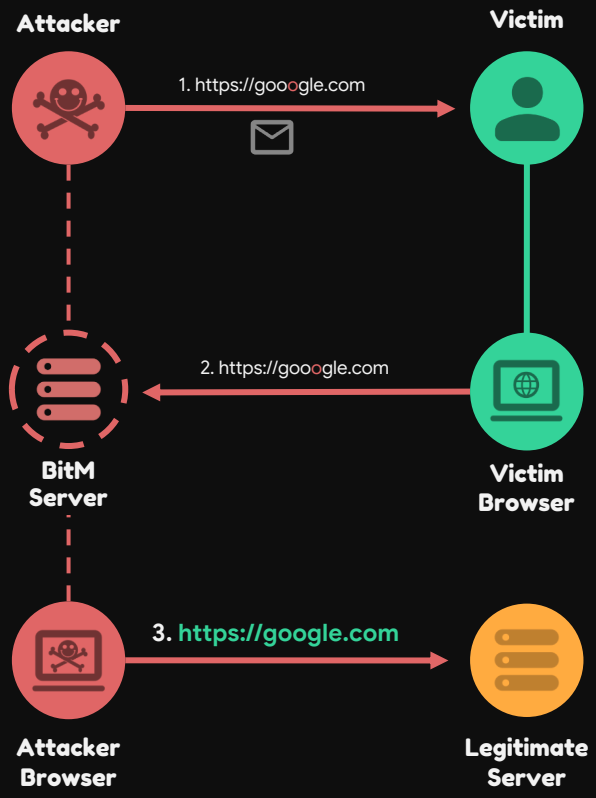
2.4 Browser-in-the-Middle

Attack Flow

1. Phishing Link Delivery

2. Remote Browser Connection: Instead of loading a fake cloned site, the victim instantly connect to a live, interactive stream of the attacker's remote browser.

3. **Direct Target Interaction:** The attacker's remote browser communicates directly with the legitimate target website.



2.5 Browser-in-the-Middle

Breaking the Hardened Web

Injections Blocked

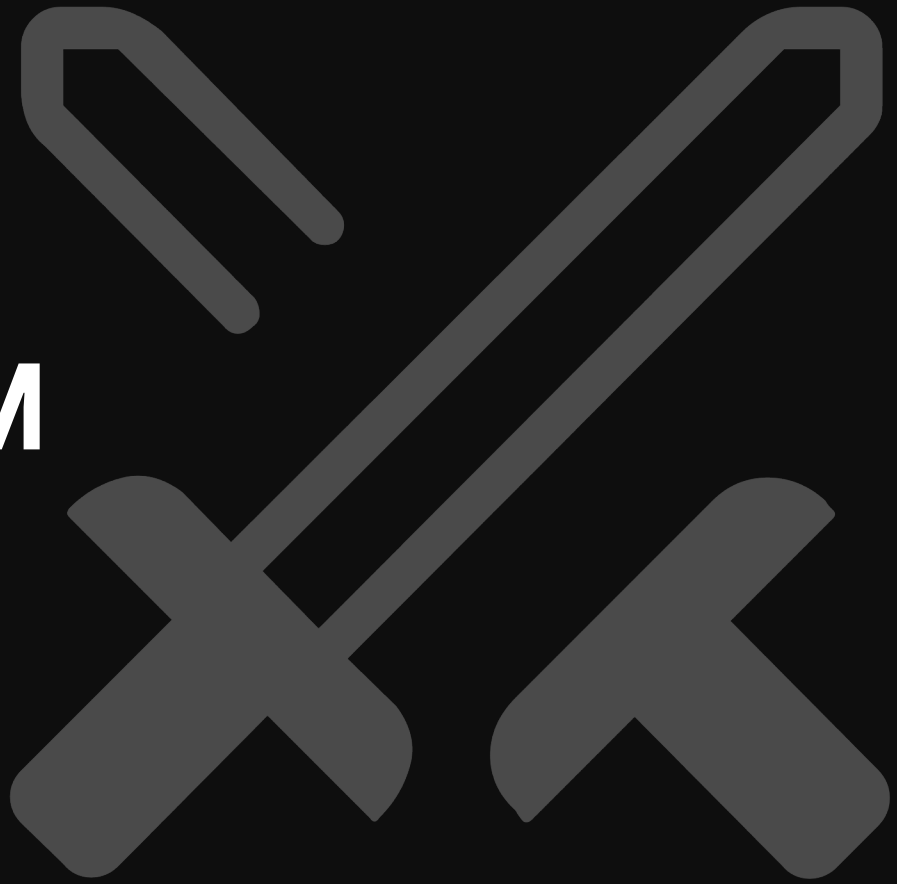
Framing Neutralized

Cookies Locked Down

3.

Reinventing BitM

A Research Journey



3.1 Reinventing BitM

The BitM Operational Gap

The Academic Blueprint

- Remote Browser Stream
- Native MFA Bypass

The PoC Bottleneck

- Single Victim
- Lack of stability

The Tactical Gap

- Non-Modular
- No Persistence

3.2 Reinventing BitM

The Engineering Problems Nobody Had Solved

Session Constraints

❌ Logout invalidates victim's session

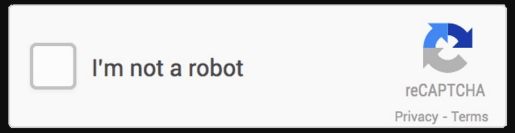
Client-side Desync

❌ Static Metadata

❌ Isolated Clipboards

Headless Browser Detection

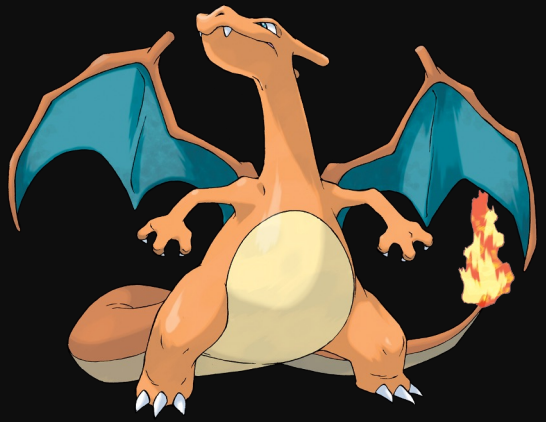
❌ Browser Fingerprints



Time to level up...

Classic Phishing

Browser-in-the-Middle



???

4.
Persistent Browser-in-the-Middle

Full Session Dominance

Challenges

- **The Foothold Challenge:** Transforming a volatile phishing session into a persistent browser-based control channel.
- **Real-Time Realism:** Eliminating interaction latency to flawlessly mirror a legitimate user experience.
- **Infrastructure Scalability**
- **Expanding the Offensive Surface:** Engineering novel attack vectors to automate and control live sessions.
  - **Weaponize Firefox extensions**
  - **Weaponize WebSocket channel**

4.1 Persistent Browser-in-the-Middle

WebRTC Streaming Layer

High-Performance Transport

- ✅ WebRTC Engine
- ✅ Low-Bandwidth Resilience

Dynamic Resolution Mapping

- ✅ On-the-Fly X11 Resizing
- ✅ Pixel-Perfect Canvas

Near-Lossless Quality

- ✅ Selkies Integration
- ✅ Zero-Delay Feedback
- ✅ Lossless Visual Stream

4.2 Persistent Browser-in-the-Middle

WebRTC Streaming Layer - Native Cursor Hijacking

BitM Selkies

Linux Cursor
(X11/Debian Style)

Default Selkies behavior leaks the BitM host OS cursor style (e.g., Linux X11).

JavaScript Hook

Runtime Hooking
Hides base64 cursor

Drops the remote cursor graphics and strips server-side artifacts.

Victim Browser

Native Alignment
CSS Local Render

Client-side CSS maps standard properties (e.g. cursor: pointer) to force local OS rendering.

4.3 Persistent Browser-in-the-Middle

WebRTC Streaming Layer - Native Cursor Hijacking

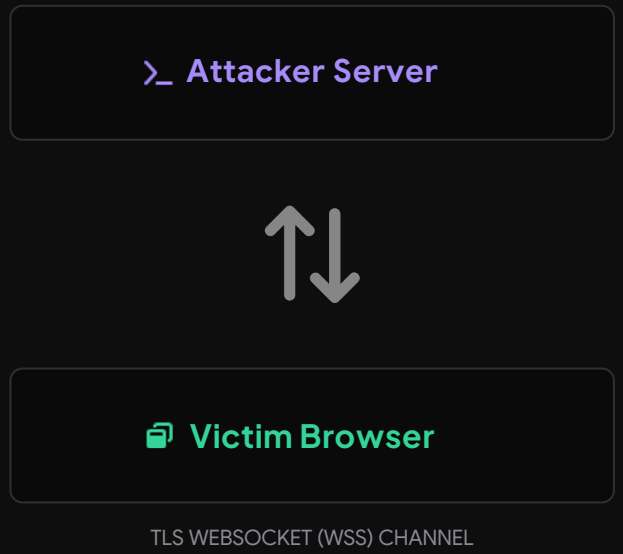
```
1  // if (window.webrtcInput && typeof window.webrtcInput.updateServerCursor === 'function') {
   //     window.webrtcInput.updateServerCursor(cursorData);
   // }

   if (cursorData.curdata) {
2    switch (cursorData.curdata) {
       case DEFAULT_CURSOR:
         overlayInput.style.cursor = cursorData.type || 'default';
         break;
       case TYPE_CURSOR:
         overlayInput.style.cursor = cursorData.type || 'text';
         break;
       case POINTER_CURSOR:
         overlayInput.style.cursor = cursorData.type || 'pointer';
         break;
       default:
         overlayInput.style.cursor = "default";
         break;
     }
   }
```

1. **Server-Side Decoupling:** Disabled the legacy server-to-client cursor update.

2. **Dynamic Client-Side Rendering:** Intercepted the victim's cursor state (*curdata*) and dynamically injects the matching CSS cursor type.

4.4 Persistent Browser-in-the-Middle

The WebSocket Control Channel



Force Download Files

Push and execute automated, silent file downloads on the target host.

Client-side Sync

Dynamically synchronizes the attacker's browser tab title and favicon with the victim's browser.

Real-time JavaScript Injection

Forces execution of arbitrary JavaScript commands directly inside the active victim's browser and manipulating DOM.

4.5 Persistent Browser-in-the-Middle

The WebSocket Control Channel - Client-side Sync

BitM
Attacker Browser

`Victim Navigation Interception`

Captures victim's navigation, tab states, and URL changes in real time.

WebSocket

`Send title/favicon`

Victim Browser

`Real-Time Client Synchronization`

Instantly forces matching tab titles and favicons on the client side.

4.6 Persistent Browser-in-the-Middle

Absolute Session Control

The BitM Advantage

Owning the execution server grants unrestricted, real-time visibility over every interaction.

Total Interaction Visibility

- **Native Keylogging**
- **Victim Session Recording:** Enables real-time screen capture and full session recording.

Operational Supremacy

- **Active Session Takeover:** Allows the operator to take the full control over the victim's session.

4.7 Persistent Browser-in-the-Middle

The Hardened Kiosk Jail

The Baseline BitM Strategy

Forcing the victim into a hardened, single-application sandbox using native browser containment.

The Lockdown

- **No Address Bar & Navigation**
- **No Window Controls**
- **No Browser Menus** (toolbars, settings, and context menus)

The Operational Impact

- **Sandbox:** Forces interaction solely with the target application, preventing the victim from exiting the browser.
- **Maximized Realism:** the session seamlessly mimics a native desktop.

4.8 Persistent Browser-in-the-Middle

The Hardened Firefox Kiosk Jail

The Advanced P-BitM Strategy

Moving beyond native Kiosk limitations to forge a surgically tailored, unbreachable browser environment.

Surgical UI Tailoring

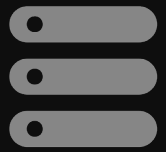
- **Direct CSS Manipulation (userChrome.css):** Create a custom ad-hoc phishing browser.

Engine Lockdown

- **Enterprise Policies:** Disable default Firefox features and block dangerous internal protocols (like `about:` and `file:///`).
- **User Preferences (user.js):** Force specific operational settings.

4.9 Persistent Browser-in-the-Middle

Weaponizing Firefox Extensions



Server-Side Vantage Point

Custom Firefox extensions

Native Core

Weaponizing standard API layers

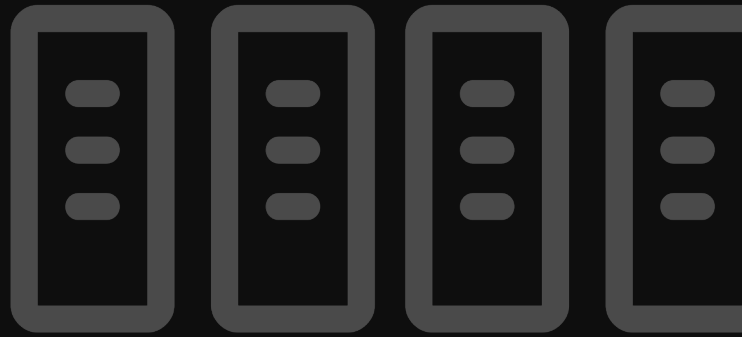
Total Traffic Governance

Act as a proxy

5.

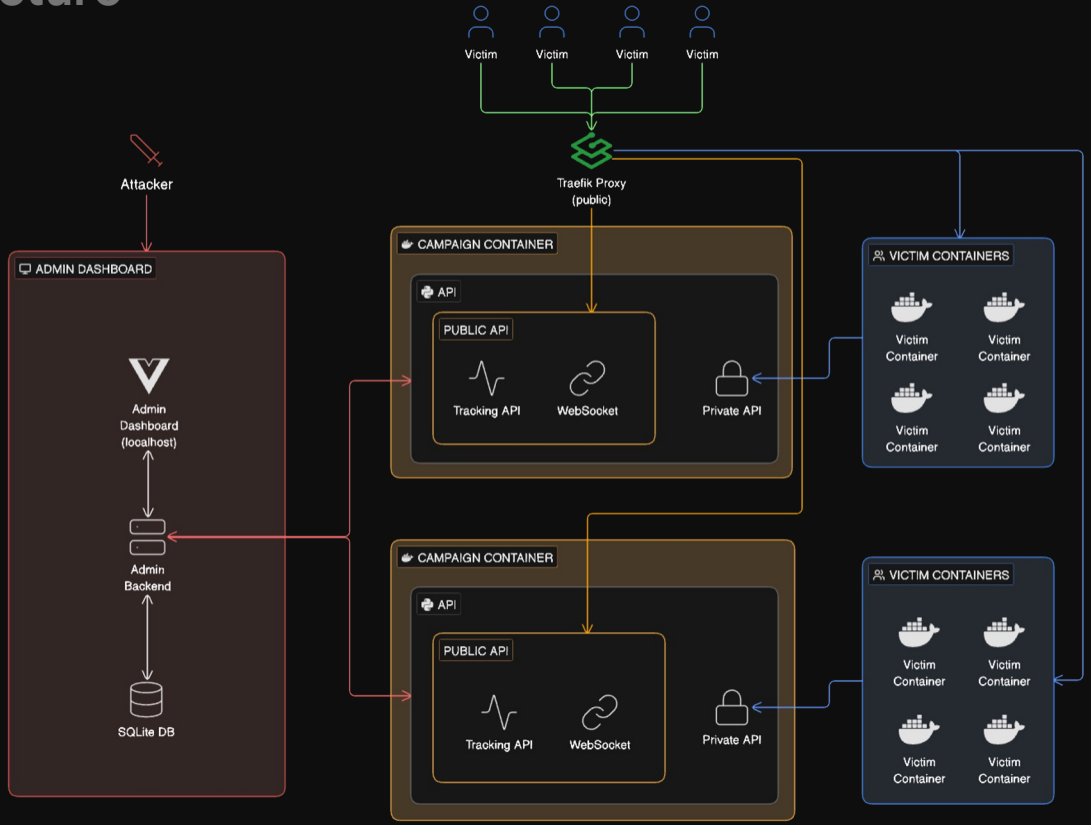
P-BitM

The First Operational Framework



5.1 P-BitM

Core Infrastructure



5.2 P-BitM

Engineering a Scalable BitM Platform

Victim Containerization

On-demand isolation: One isolated Docker container per target campaign and victim.

Zero crosstalk: Dedicated isolated browser with its own session state.

APIs

REST: Campaign orchestration and backend state automation.

Data Channels: Private endpoints to manage exfiltrated session state.

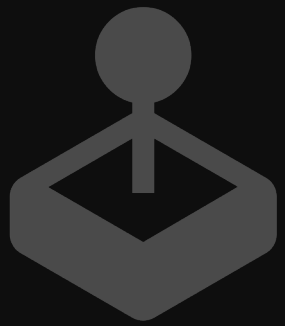
WebRTC Streaming Layer

Selkies WebRTC: High-performance real-time transport layer.

Near-lossless latency: Near-lossless visual stream for long-running sessions.

6.
Operator Console & Live Session Control

Puppeteering Live Targets



6.1 Operator Console & Live Session Control

Gophish-style Campaign Management

1

Target & Scheduling

Define user groups, target profiles, and automated launch windows for targeted campaigns.

2

DNS & SMTP

Handle automated email delivery to initial targets using custom SMTP profiles and dedicated domains.

3

Tracking

Continuous monitoring of email delivery, open rates, and malicious link interaction.

4

Target Domain Binding

The platform spins up a headless browser instance and shares the remote desktop to the victim.

6.2 Operator Console & Live Session Control

Admin Dashboard

Campaign Management

Streamlines the creation, deployment, and tracking of targeted phishing operations from a centralized interface.

Real-Time Session Control

Provides live monitoring and direct interaction with active victim browser sessions as they happen.

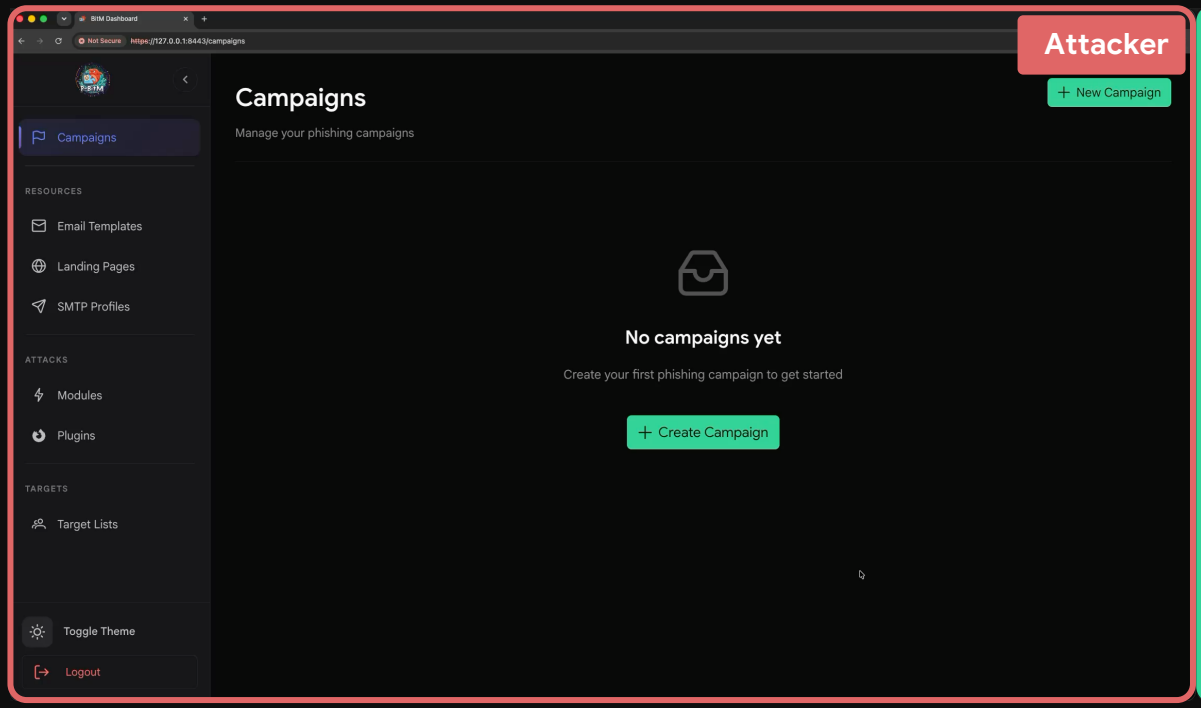
Modular Extension Engine

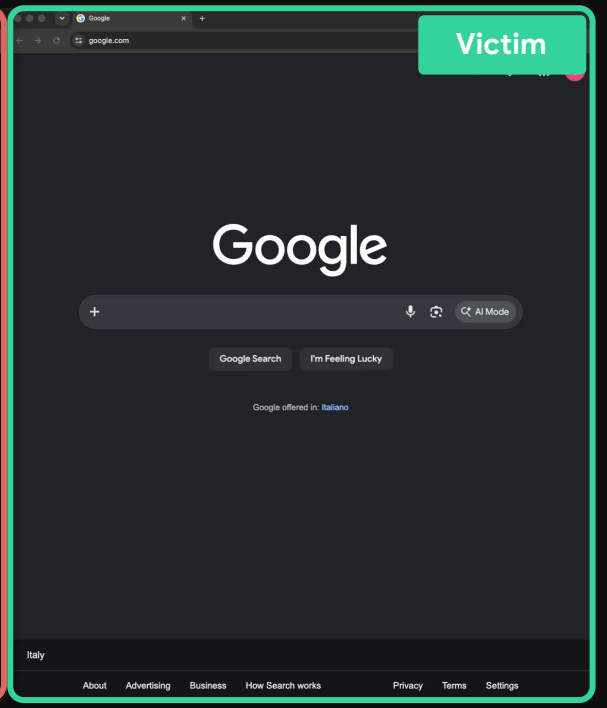
Create custom Firefox extensions and tailored client-side modules to expand operational capabilities.


Time to show what P-BitM
can actually do...

6.3 Operator Console & Live Session Control

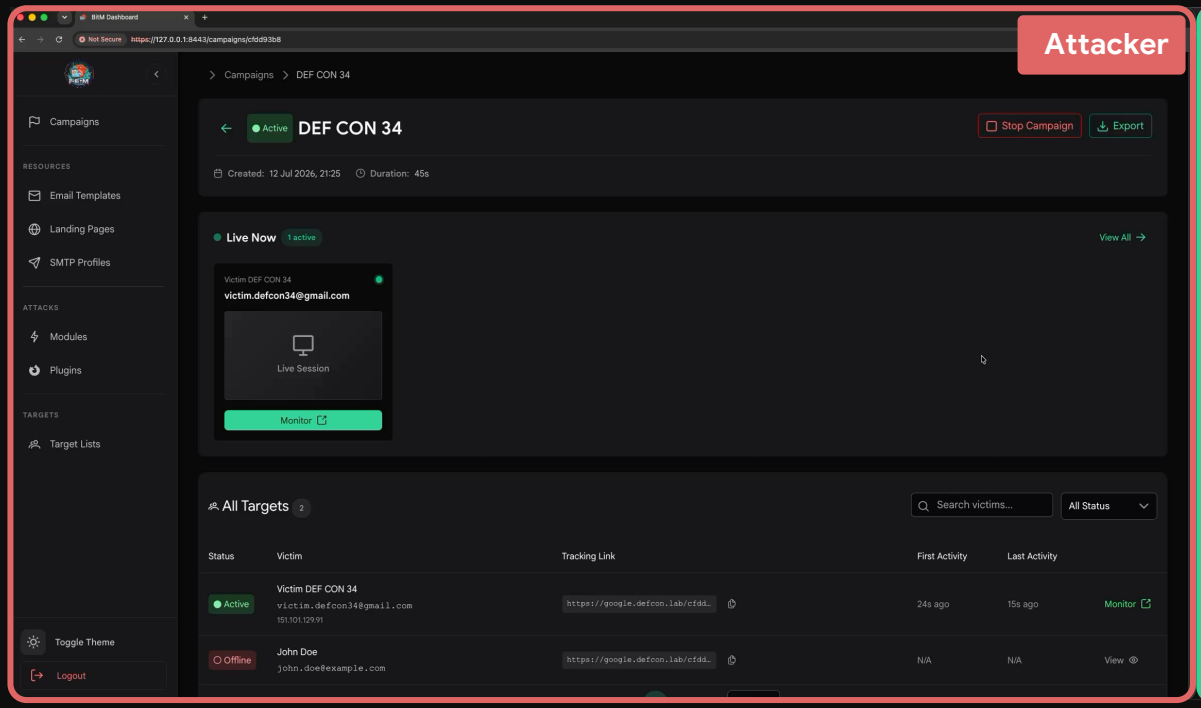
Admin Dashboard - Campaign Creation + MFA Bypass – DEMO 1

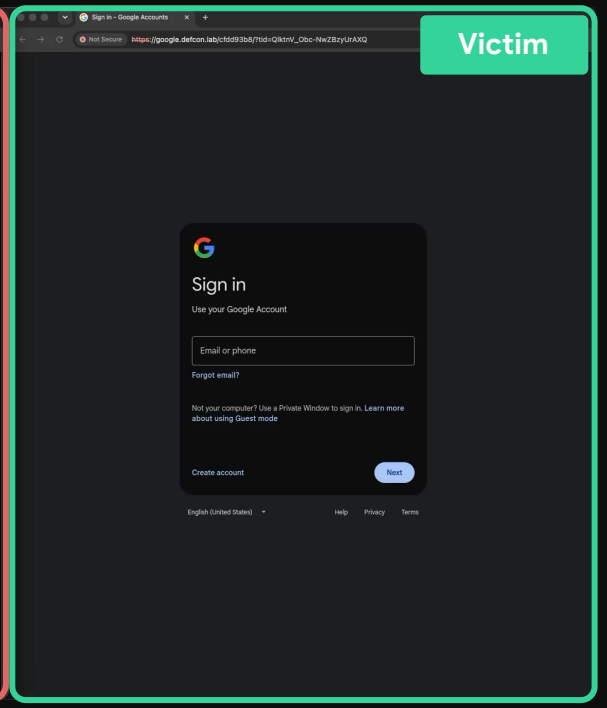




6.4 Operator Console & Live Session Control

Admin Dashboard - Real-Time Monitoring – DEMO 2



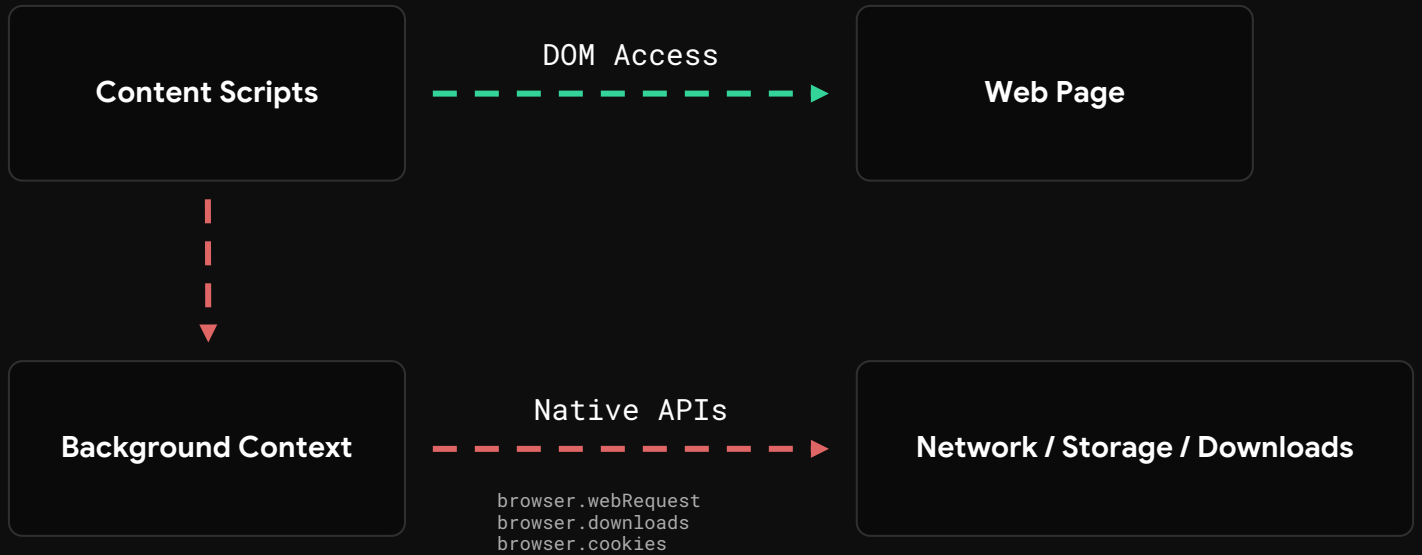


7.
Weaponizing Firefox Extensions

Native API Exploitation

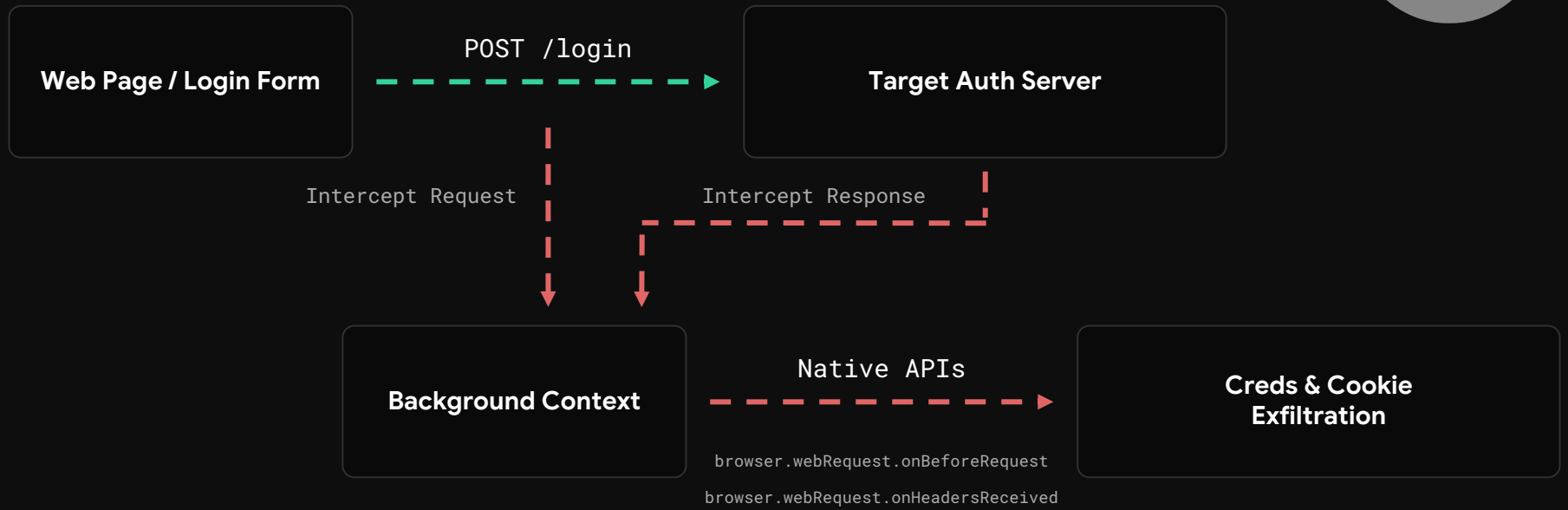
7.1 Weaponizing Firefox Extensions

The Architecture of Firefox Extensions



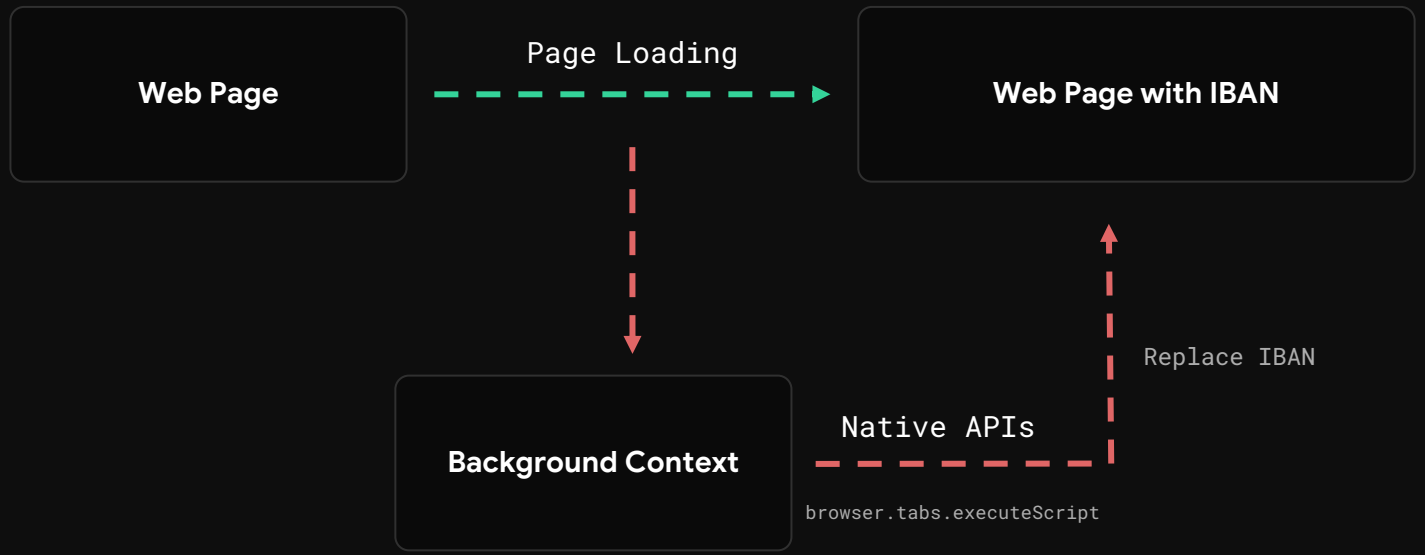
7.2 Weaponizing Firefox Extensions

Dynamic Credential Interception & Cookie Harvesting



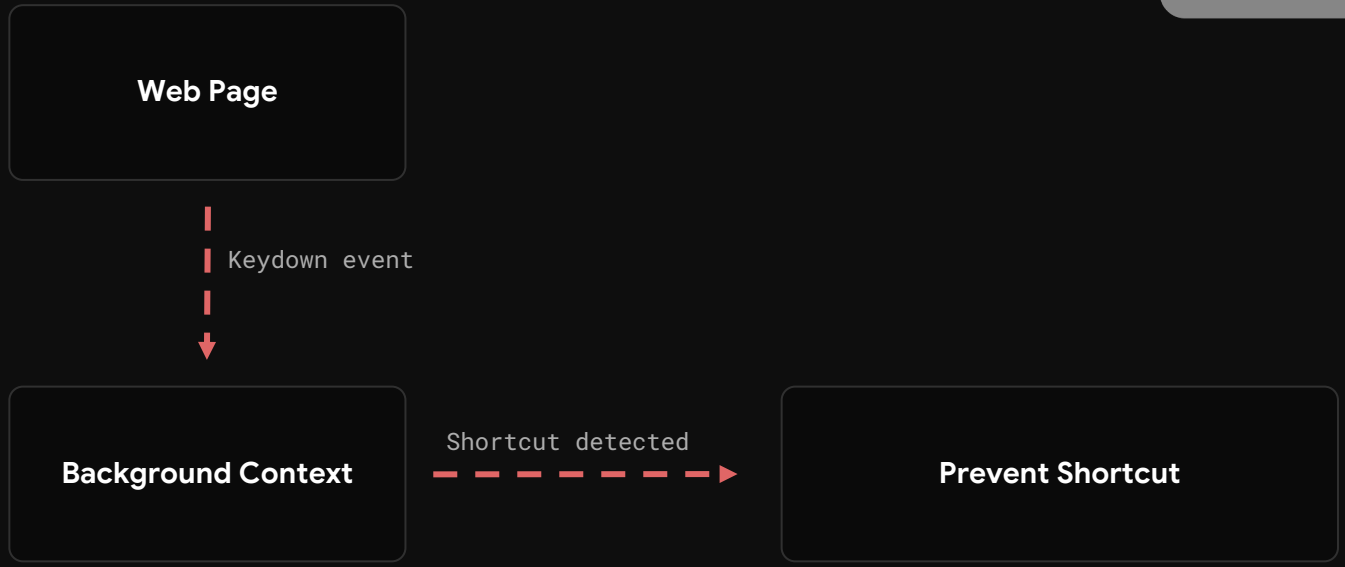
7.3 Weaponizing Firefox Extensions

In-Flight Data Manipulation



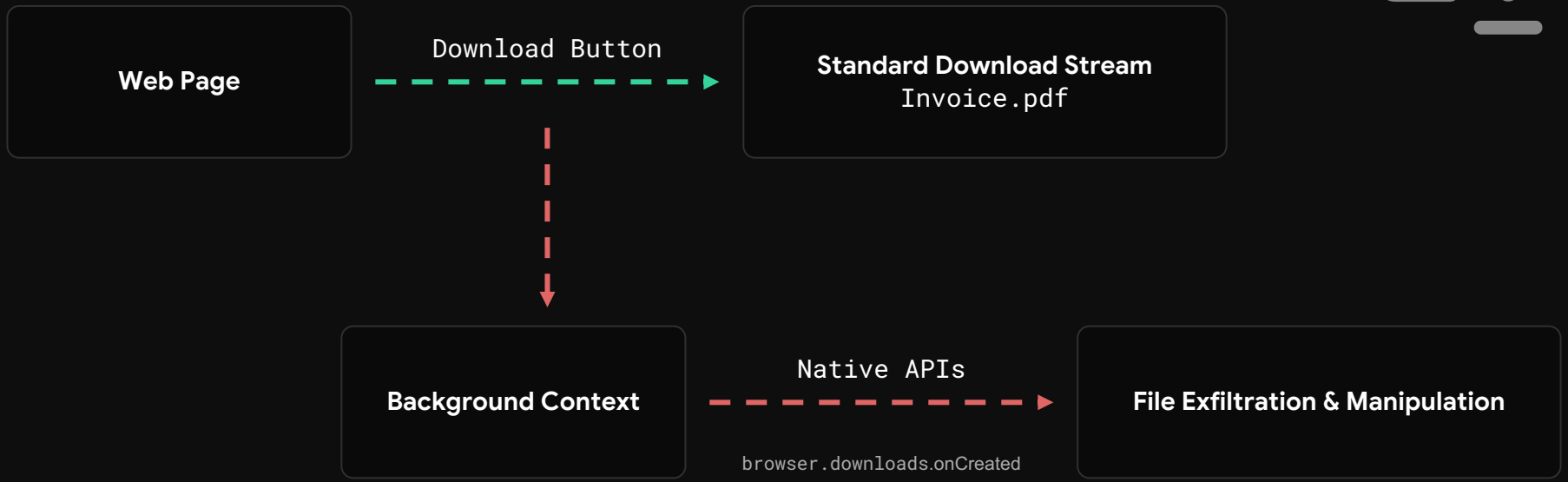
7.4 Weaponizing Firefox Extensions

Preventing Sandbox Bypasses by Restricting Firefox Shortcuts



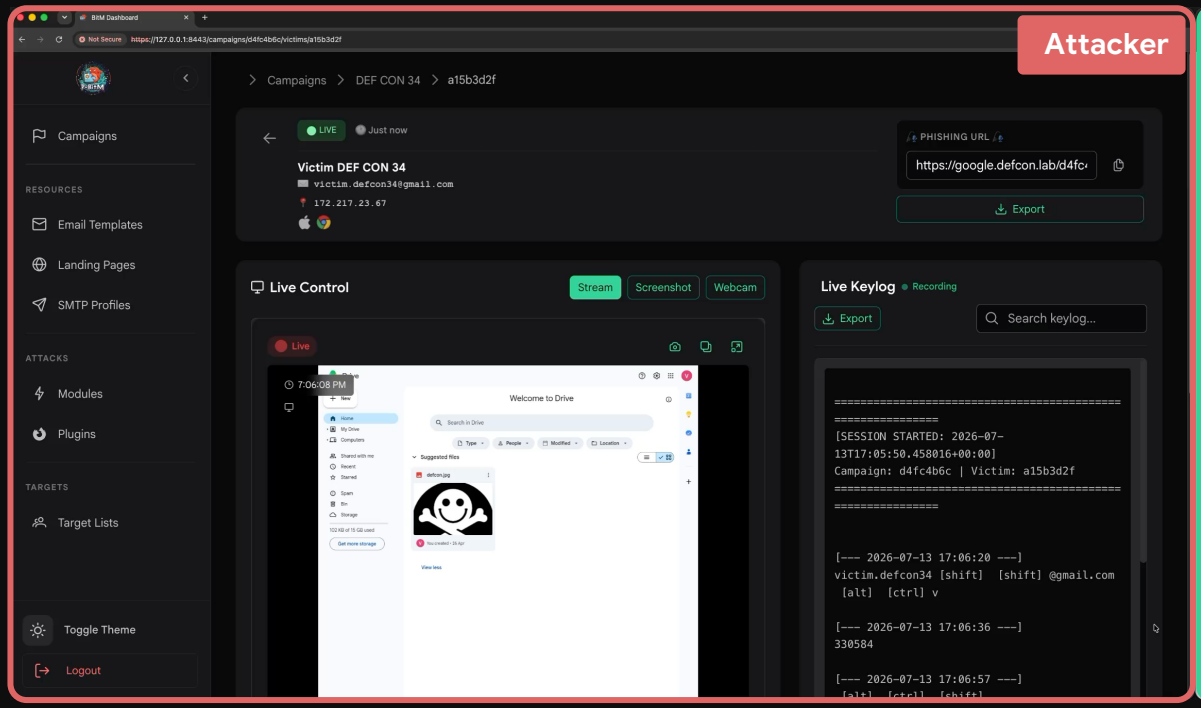
7.5 Weaponizing Firefox Extensions

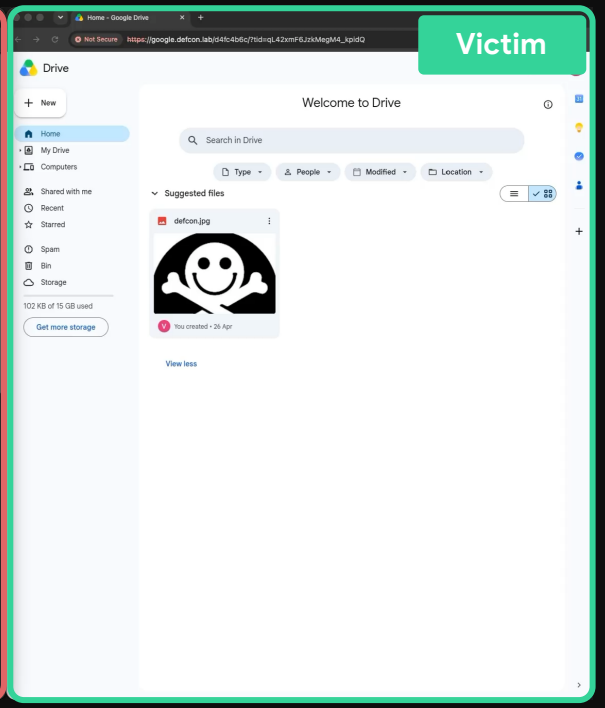
File Hijacking & Download Interception



7.6 Weaponizing Firefox Extensions

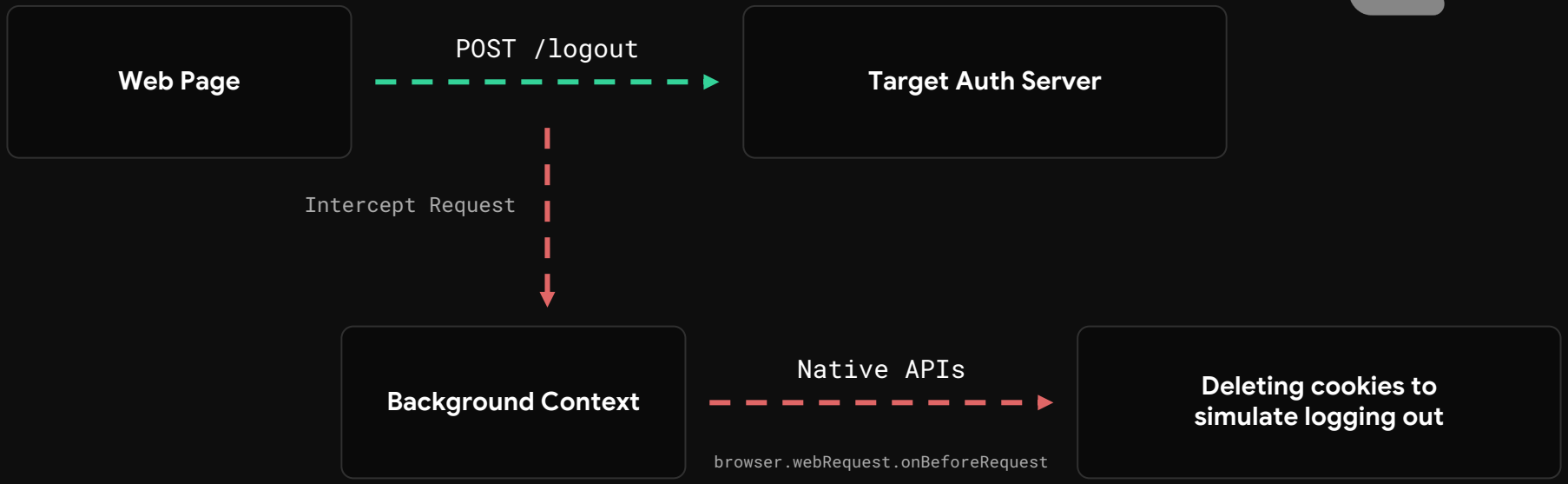
File Hijacking & Download Interception – DEMO 3





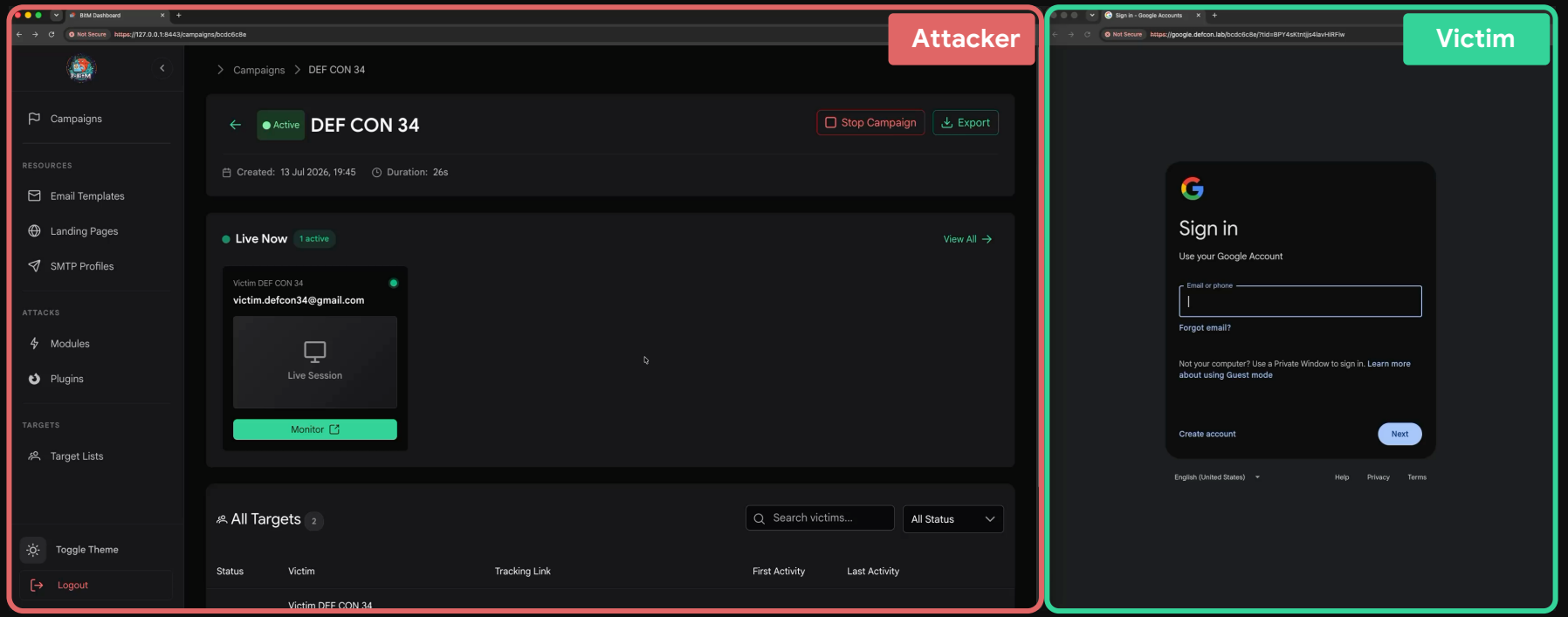
7.7 Weaponizing Firefox Extensions

Persistence Logout



7.8 Weaponizing Firefox Extensions

Persistence Logout – DEMO 4



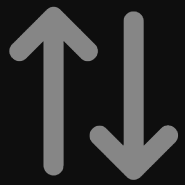
8.

Hook like a BeEF

Arbitrary JavaScript Execution via WebSocket Streams

8.1 Hooking like a BeEF

Old but Gold



Persistent WebSocket Channel

Establishes a WebSocket channel allowing the operator to push dynamic, unconstrained JavaScript payloads directly into the victim's browser context in real time.

Hardware & Environment Access

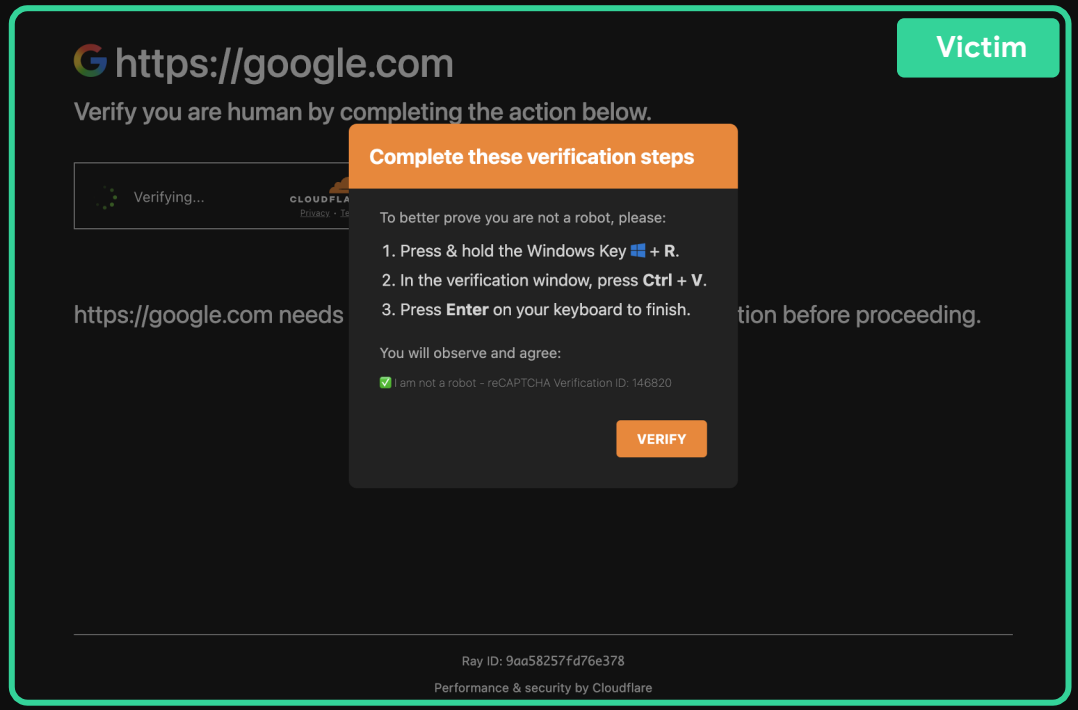
Abuses built-in HTML5 capabilities and native browser APIs to capture live media streams and harvest host environmental metadata.

8.2 Hooking like a BeEF

The ClickFix Technique

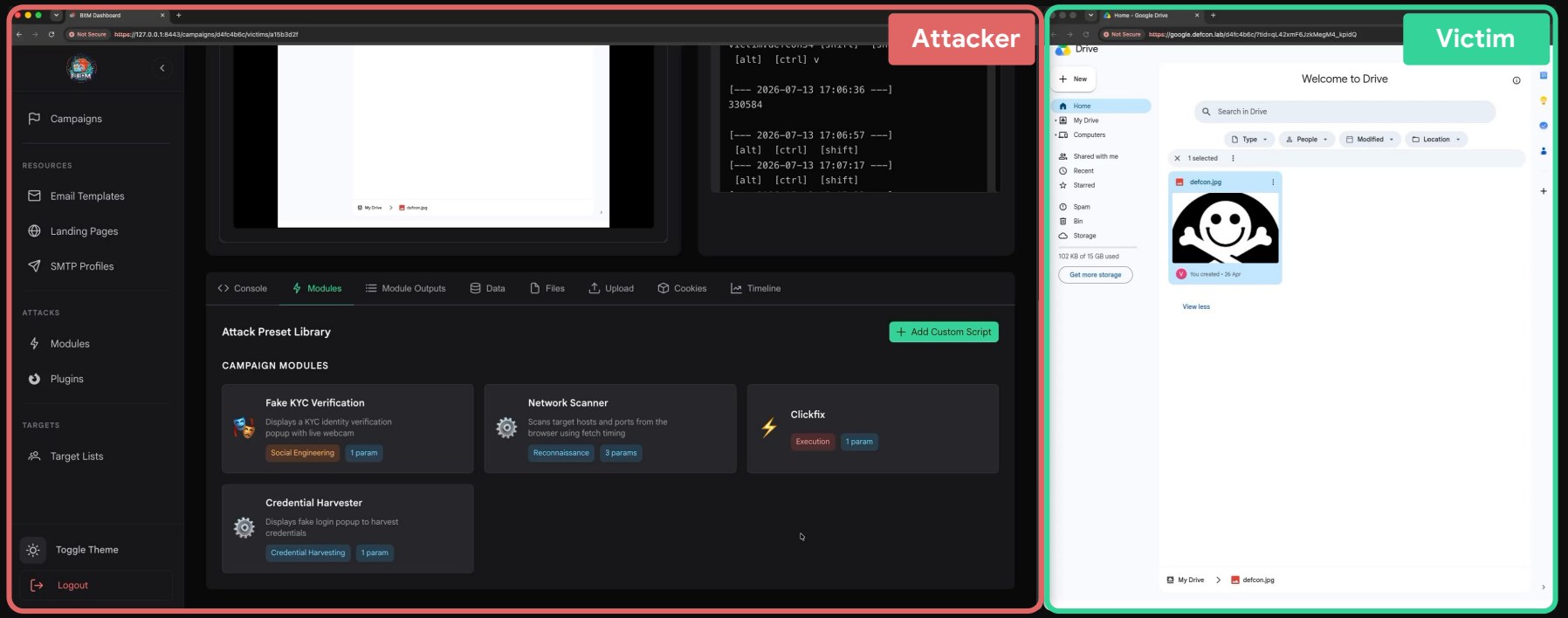
Dynamic DOM Manipulation: Overlaying the current page layout to inject highly realistic, context-aware error prompts or fake update alerts.

Social Engineering Delivery: Forcing the browser to display urgent operational instructions that prompt the user to execute malicious commands on their host system.



8.3 Hooking like a BeEF

The ClickFix Technique – DEMO 5

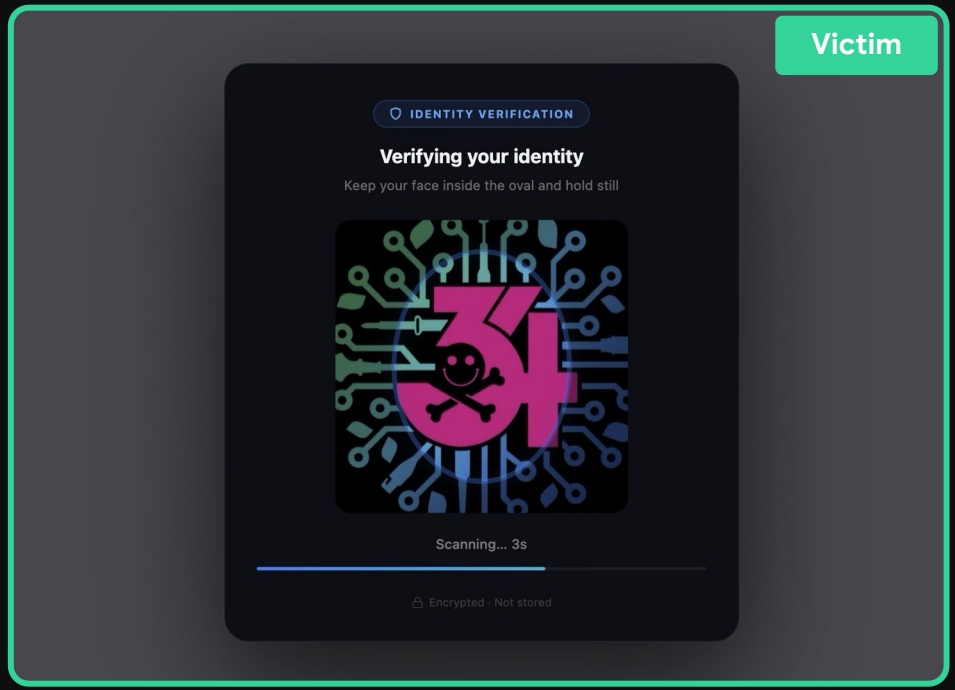


8.4 Hooking like a BeEF

The Fake KYC Bait

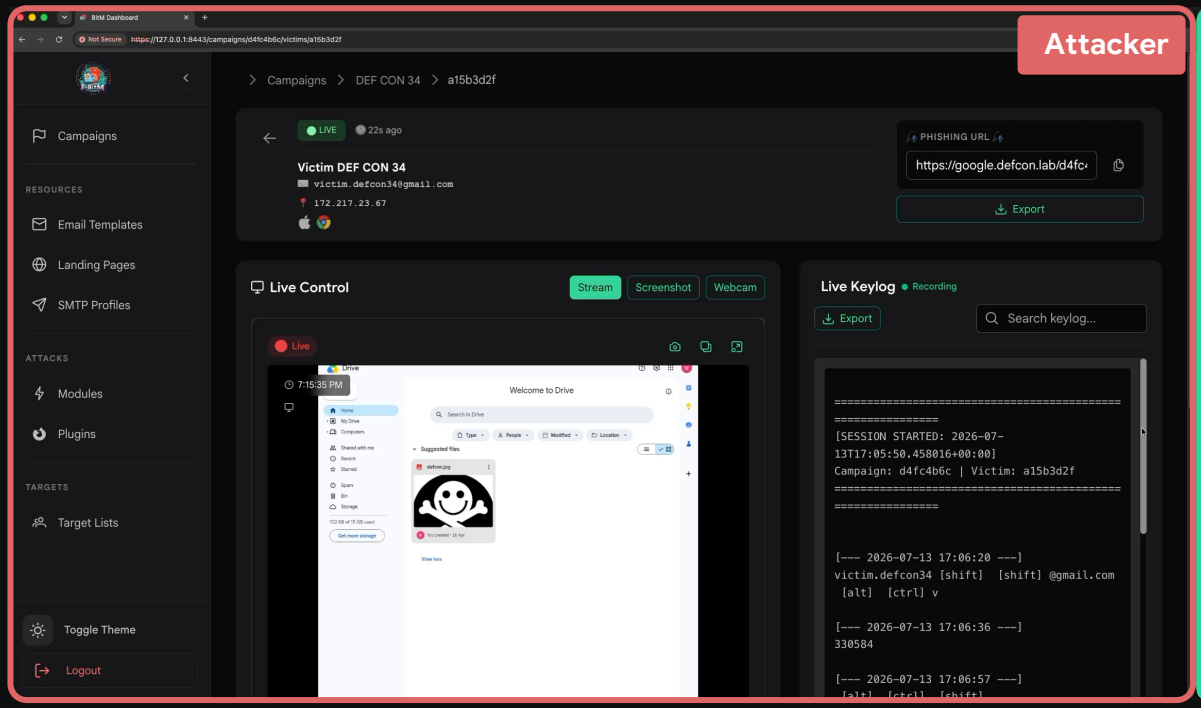
Real-Time Identity Harvesting:
Injecting a rogue, convincing Know Your Customer (KYC) verification overlay to trick the user into allowing access to the webcam and microphone.

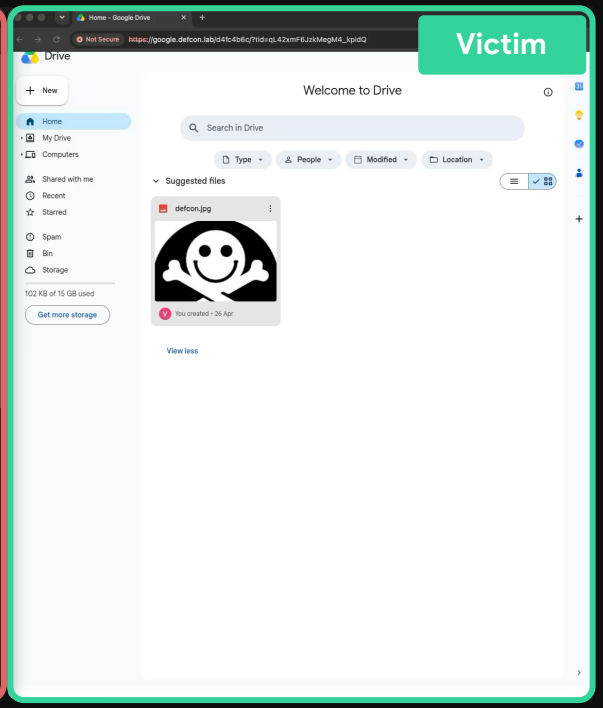
Hardware API Hijacking:
Programmatically invoking and controlling the victim's webcam and media streaming capabilities directly through the WebSocket channel.



8.5 Hooking like a BeEF

The Fake KYC Bait – DEMO 6



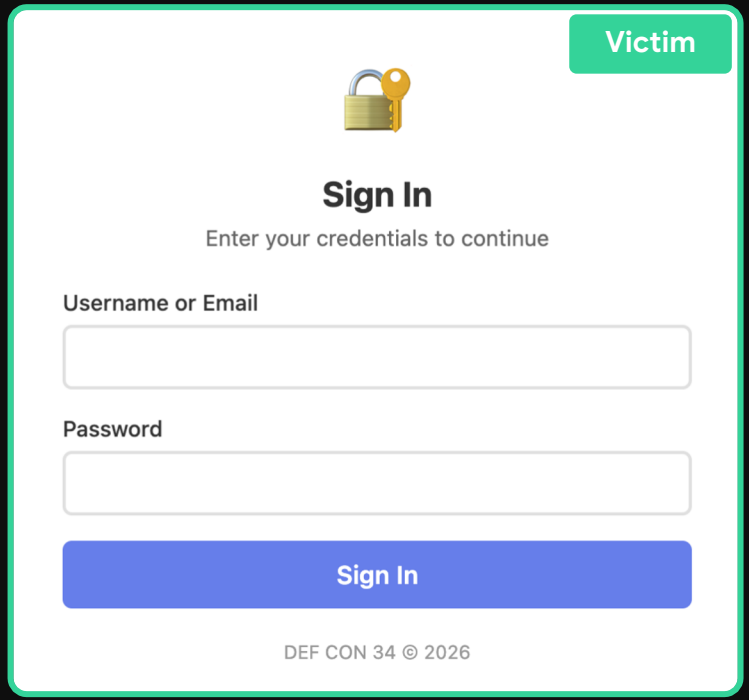


8.6 Hooking like a BeEF

Inline Injected Login Forms

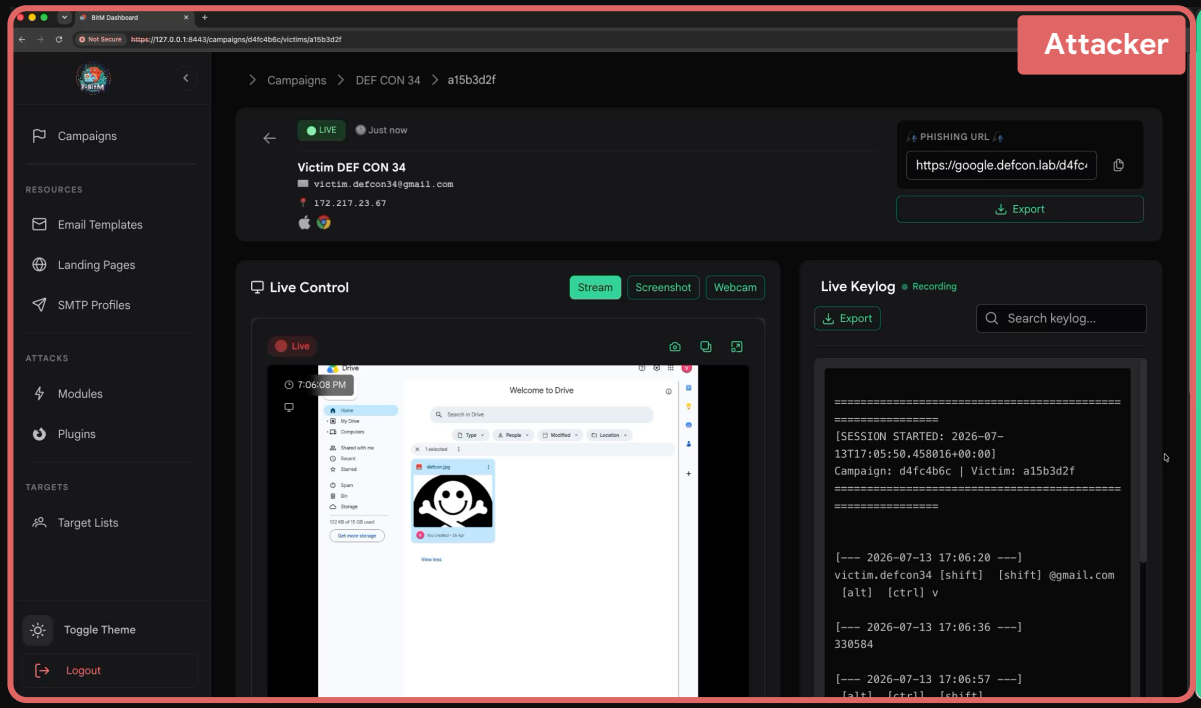
Phishing-in-the-Middle:
Injecting completely rogue authentication prompts or fake session-expired overlays directly over legitimate web apps.

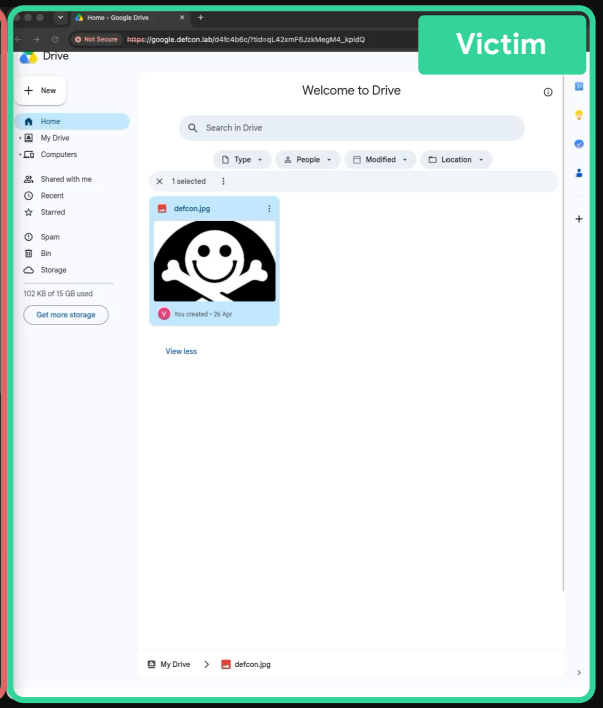
Credential Harvesting:
Capturing username, password, and other sensitive contextual information in real time as the user interacts with the page.



8.7 Hooking like a BeEF

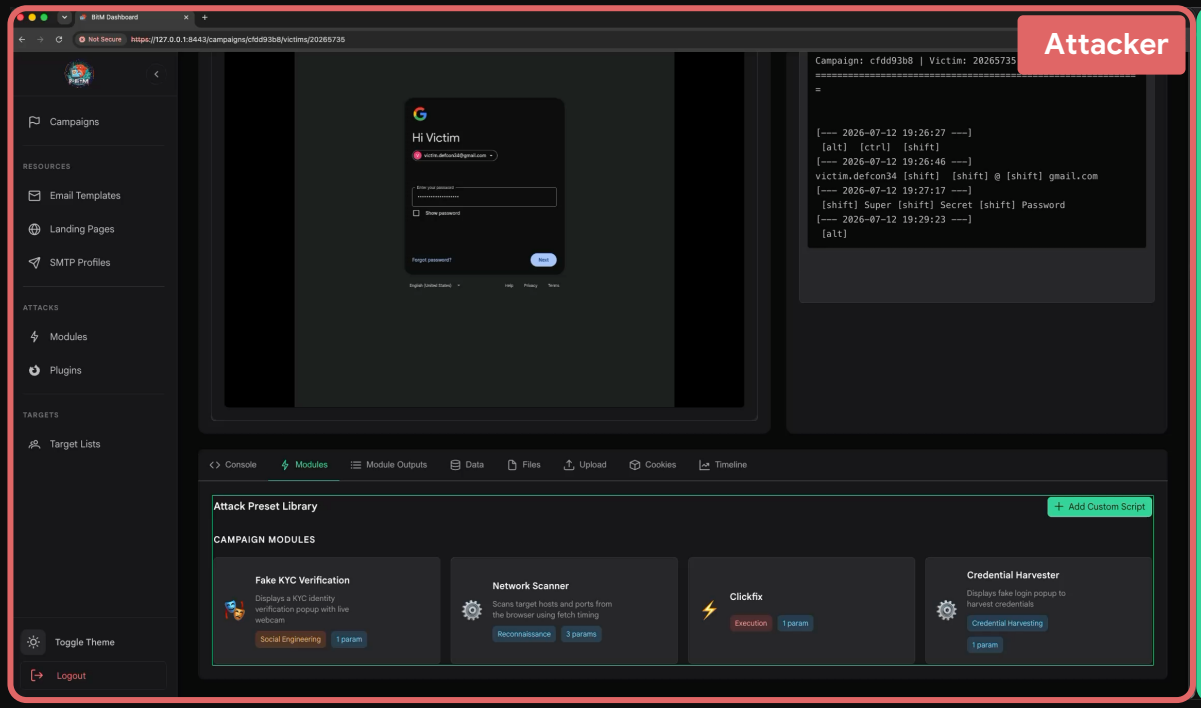
Inline Injected Login Forms – DEMO 7

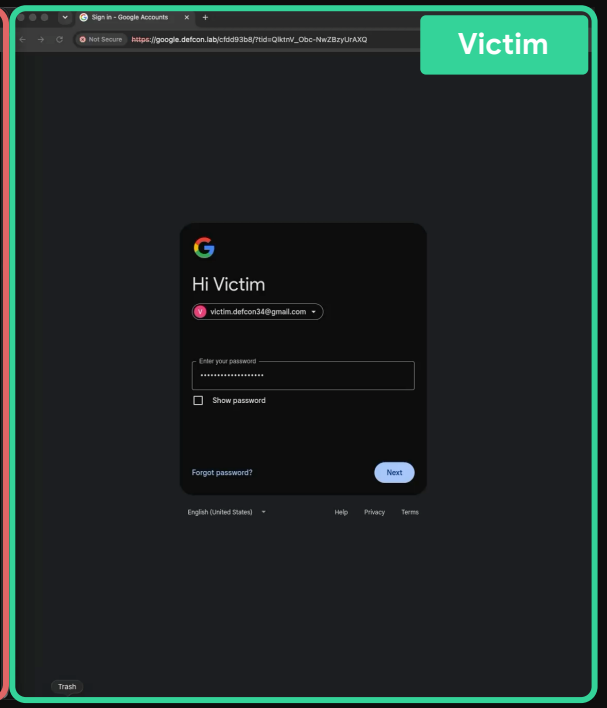




8.8 Hooking like a BeEF

Network Scanner- DEMO 8





9.

Blue Teaming P-BitM

Mitigating, Detecting, and Breaking BitM Frameworks

9.1 Blue Teaming P-BitM

Identifying Indicators of Compromise and Behavioral Anomalies

Anomalous Protocol Behavior

Flagging unexpected WebRTC session establishments originating from arbitrary web contexts and detecting persistent WebSocket traffic patterns.

Geolocation Telemetry

Analyzing authenticated sessions originating from anomalous IP addresses immediately following a phishing interaction, and alerting on geographically inconsistent access patterns.

Client-Side Content Integrity

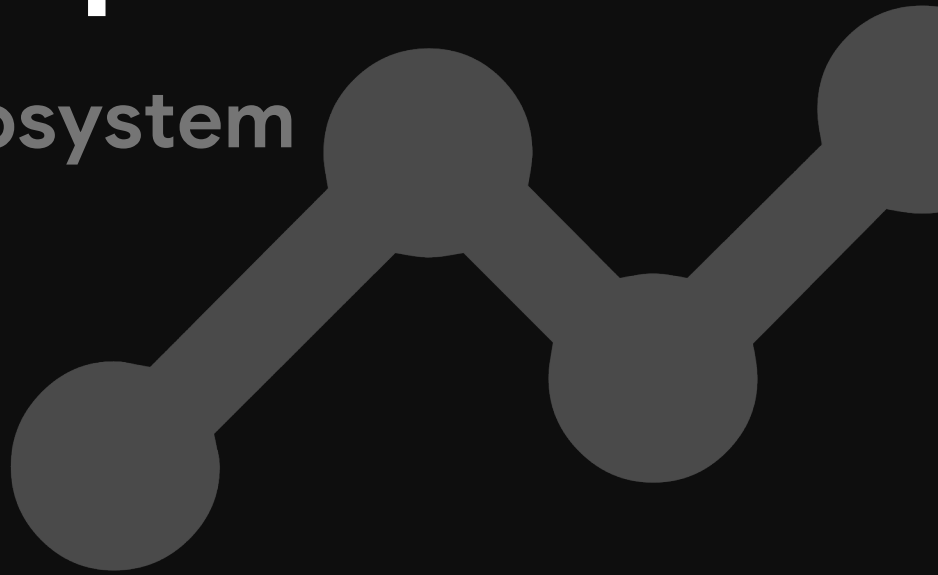
Monitoring high-frequency JavaScript DOM mutation events indicative of modular injection frameworks, and detecting file hash mismatches resulting from in-transit download modifications.

Intentional IoCs

Enforcing actively hunting for P-BitM's hardcoded infrastructure fingerprint embedded across all responses.

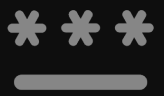
10.
Future Roadmap

Evolving the P-BitM Ecosystem



10.1 Future Roadmap

Evolving the Persistent Browser-in-the-Middle Ecosystem



Password Manager Spoofing

Chromium-based Support

AI-Driven Phishing Flows

Community-Driven Modules

Open Source Core Release

Empowering the Community with Deployable Frameworks

Open Source Core Release

Evolving the Persistent Browser-in-the-Middle Ecosystem

A core version of P-BitM designed to be immediately deployable and community-extensible will be released as open-source after DEF CON 34

- **Core Framework**
- **Base Modules & Firefox Extension**
- **Documentation**

```
$ git clone https://github.com/P-BitM-Framework/P-BitM
```

Thanks DEF CON!

Giacomo Lenzini

GiacoLenzo2109@proton.me

giacomo-lenzini