Securing SaaS–Merchant Connections

Overview

Securing data between merchant websites and SaaS servers is vital. While we can enforce robust measures on our end, complete control over an end-user's browser isn't possible.

Key Responsibilities

- **SaaS Providers:**
  - Enforce SSL/TLS for all data transmissions.
  - Use secure cookies (Secure, HttpOnly, SameSite).
  - Conduct regular security audits and enforce multi-factor authentication.
- **Merchant Websites:**
  - Ensure all site traffic is HTTPS-only.
  - Implement best practices like Content Security Policies (CSP) to guard against attacks.
- **End-Users:**
  - Maintain updated devices and browsers.
  - Understand that security on their end ultimately depends on their own practices.

Case Example: HTTPS Interception with mitmproxy

- **Creating a Python-Based Proxy Server (server1):**
  - Ran **mitmproxy** on `localhost:8080` to serve as your proxy server (server1).
- **Routing All Traffic Through the Proxy:**
  - Configured your MacBook's network settings so that all HTTP/HTTPS traffic is directed to `localhost:8080`.
  - **What This Means:**
    All browser connections that were previously made directly to the target server (chargebee.com) are now routed to *server1 (the proxy). The browser sends its requests to server1, and server1 forwards them to chargebee.com.*
- **SSL Certificate Requirement:**
  - Since server1 acts as an intermediary, it must terminate (decrypt) the HTTPS connections coming from the browser.
  - **Decrypted Data From:**
    The browser's encrypted HTTPS data is decrypted at server1.
  - **Secure Connection With:**
    Server1 establishes its own secure TLS session with chargebee.com.
  - **Important Note:**
    Server1 uses its own certificate (cert2) for the connection with the browser. To allow the browser to trust this connection, you must install server1's (mitmproxy's) certificate on your MacBook. (Refer to the next step.)
- **Installing and Trusting the mitmproxy Certificate:**
  - mitmproxy provides its own self-signed proxy certificate.
  - Downloaded the certificate (via http://mitm.it) and added it to the Apple Keychain.
  - Set the certificate's trust level to "Always Trust" so that macOS accepts it without warnings.
- **Two Separate TLS Sessions:**
  - **Browser ↔ server1 (Proxy):**
    - The browser establishes a secure TLS session with server1.
    - Server1 presents its own certificate (cert2) to the browser. (Remember: you must install this certificate on your MacBook so that it is trusted.)

        - This allows server1 to decrypt the HTTPS data sent by the browser.
    - **server1 (Proxy) ↔ chargebee.com (Target Server):**
        - Server1 connects to chargebee.com, which sends its own certificate (cert1).
        - A separate secure TLS session is established between server1 and chargebee.com.
- **Intercepting and Modifying Requests/Responses:**
    - With all traffic routed through server1, mitmproxy intercepts every request.
    - It decrypts the browser's request data (using the trusted cert2) and forwards it to chargebee.com.
    - The response from chargebee.com is decrypted by server1 and can be inspected or modified (for example, changing GTM tag data) before being re-encrypted with cert2 and sent back to the browser.
- **Final Outcome:**
    - A secure connection is maintained between your browser and server1, and separately between server1 and chargebee.com.
    - Server1 now sits in the middle, intercepting and modifying traffic as required.
    - **Done!**

Before Adding Proxy

After Adding Proxy Server

Proxy Server -