Resumen Semana 12

Durante estas dos próximas semanas, el reto consiste en aprender a usar Burp Suite y vulnerar la Juice Shop de TryHackMe.

Esta semana vamos a ver qué se necesita para empezar a hackear WebApps.

HTTP

Vamos a revisar el protocolo HyperText Transfer Protocol (HTTP). Este protocolo se inventó para servir datos (contenido web, html, css, js…) desde un servidor a los clientes, para que fuese visualizado por un Navegador Web.

Veamos como se realiza una petición HTTP a un servidor. Coloquemos en la URL https://www.twitch.tv/alexelcapo y presionamos Enter. Esta es la petición:

```
GET https://www.twitch.tv/alexelcapo HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Host: www.twitch.tv
```

- **Verbo, recurso y versión**
  - La primera línea del protocolo siempre corresponde con un verbo HTTP, **GET** en este caso, un recurso **https://www.twitch.tv/alexelcapo**, y la versión **HTTP/1.1**
  - Aquí podéis ver una lista de verbos HTTP (HTTP request methods - HTTP | MDN (mozilla.org))
- **Headers/Cabeceras**
  - A partir de esa primera línea vienen cabeceras, que simplemente son información adicional sobre nuestra petición
    - User-Agent indica el navegador desde el que se ha realizado la petición
    - Accept indica el tipo de datos que pretende recibir el cliente
    - Connection indica si la conexión quiere quedar abierta cuando termine la petición
    - … (https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers)

Una vez realizada esa petición, obtenemos una respuesta por parte de Twitch.

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/html
Set-Cookie: unique_id=Zaewl5QeeREs8qOxGXRBki7toCIu6bSH; expires=Thu, 05 Jan 2023 10:57:52 GMT; domain=.twitch.tv; path=/; secure; samesite=none
Set-Cookie: unique_id_durable=Zaewl5QeeREs8qOxGXRBki7toCIu6bSH; expires=Thu, 05 Jan 2023 10:57:52 GMT; domain=.twitch.tv; path=/; secure; samesite=none; httponly
Set-Cookie: server_session_id=a3a8e57b77c44a99ac078c67278da96d; domain=.twitch.tv; path=/; secure; samesite=none
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000
Cache-Control: private, max-age=0
Date: Sun, 05 Dec 2021 10:57:52 GMT
Set-Cookie: twitch.lohp.countryCode=ES; domain=.twitch.tv; expires=Fri, 30 Dec 2022 10:57:52 GMT;
Vary: Accept-Encoding
```

La respuesta es algo similar a la petición.

- **Versión y status code**
  - Ahora recibimos un código/número que nos indica como ha ido la petición, en nuestro caso **200 OK**
  - Aquí se puede ver un listado de códigos (HTTP response status codes - HTTP | MDN (mozilla.org)) En general:
    - Informational responses (100–199)
    - Successful responses (200–299)
    - Redirection messages (300–399)
    - Client error responses (400–499)
    - Server error responses (500–599)
- **Headers/Cabeceras**
  - Igual que los de la petición.

Con esta respuesta, a su vez recibimos todos los datos y ficheros necesarios para que nuestro navegador cargue la página de Twitch con el perfil de Alexelcapo.

Cookies

Todos hemos oído hablar del concepto de Cookie. Las Cookies son variables que se quedan guardadas en el navegador y se suelen mandar con cada petición HTTP que hacemos.

Las Cookies son MUY útiles porque HTTP es "**stateless**", o dicho de otra manera, cada petición es independiente. Esto quiere decir que si visitamos Twitch (sin Cookies), y una hora después lo volvemos a visitar, no hay nada que guarde nuestra sesión (como si fuera la primera vez que lo visitamos).

Por esto, las Cookies se usan generalmente para guardar sesiones y rastrear ciertos datos (para poder elegir anuncios más acertados, por ejemplo).

La información que nos interesa sobre las Cookies es el **nombre** y el **valor** de estas. Un ejemplo de Cookie es: `username=Merk` o `id=21367`

Las webs que visitamos almacenan Cookies en nuestro navegador usando la cabecera **Set-Cookie: <cookie-name>=<cookie-value>** en la respuesta a una petición.

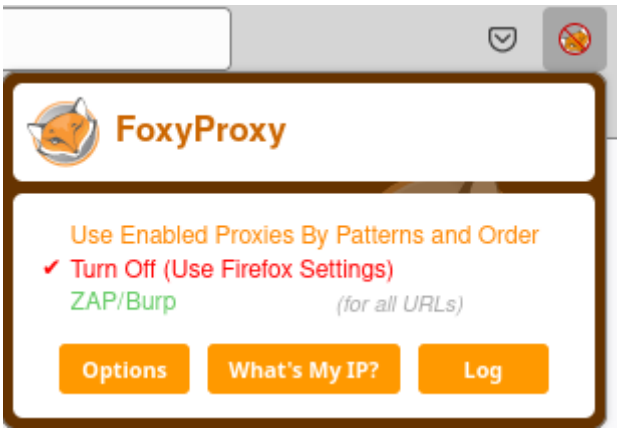
Después, todas las peticiones que realicemos a esa web tendrán la cabecera **Cookie: name=value**.

Web Proxy

Ahora que conocemos algunos de los peldaños del protocolo HTTP, vamos a conocer una herramienta muy potente que nos permite inspeccionar cada petición y respuesta enviada por nuestro navegador. Esta herramienta es **Burp Suite**.

Burp Suite viene instalado en Kali, pero para empezar a usarlo hay que configurarlo.

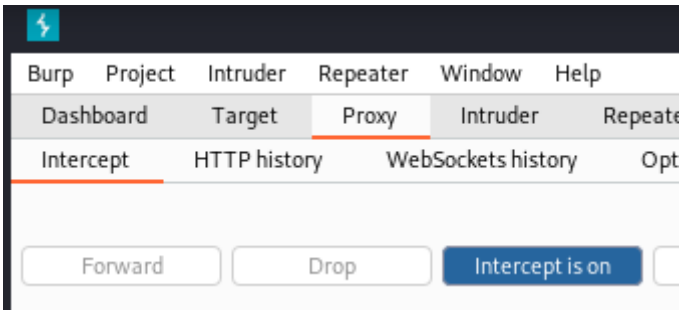
Recomiendo usar una extensión llamada FoxyProxy para hacer las cosas más sencillas. Instalamos la extensión, la pulsamos y veremos esto.



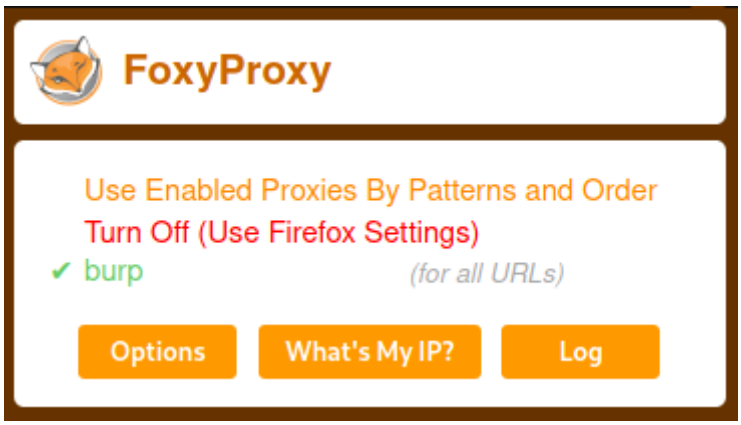
En Options -> Add, se le da un nombre y lo único que hay que cambiar es esto:

Y por último habrá que darle a Firefox los certificados de Burp Suite para poder interceptar peticiones HTTPs. Se explica muy bien cómo aquí (Installing Burp's CA certificate in Firefox - PortSwigger).

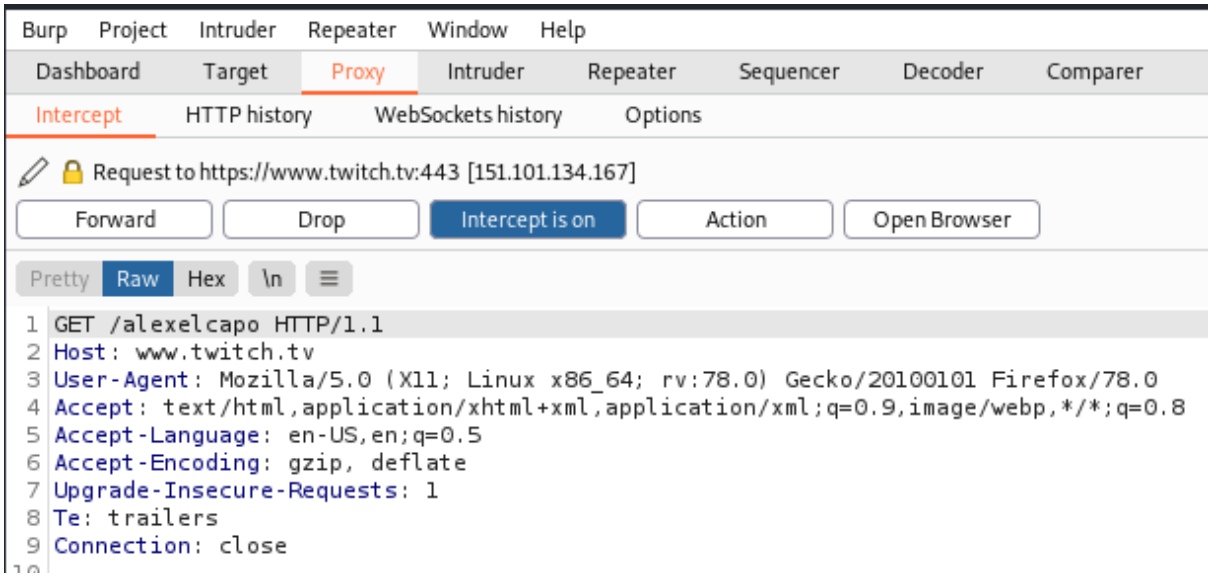
Una vez realizados estos pasos, si tenemos Burp Suite abierto en la pestaña Proxy-Intercept y marcado el **Intercept is on**…



Y activamos la configuración que hemos creado en FoxyProxy desde Firefox, todo nuestro tráfico HTTP será interceptado.



Y ahora tendremos control sobre todas nuestras peticiones en Internet, una por una.



No me voy a centrar en introducir conceptos de vulnerabilidades web como SQL Injection, Data Exposure, XSS… en este documento, porque el reto de estas semanas hace un walkthrough muy bien explicado.

Reto

[TryHackMe | OWASP Juice Shop]

Por ahora eso es todo, cualquier duda no dudéis en preguntar a cualquier compañero o a Llamas y Merk!

Hasta la semana que viene!