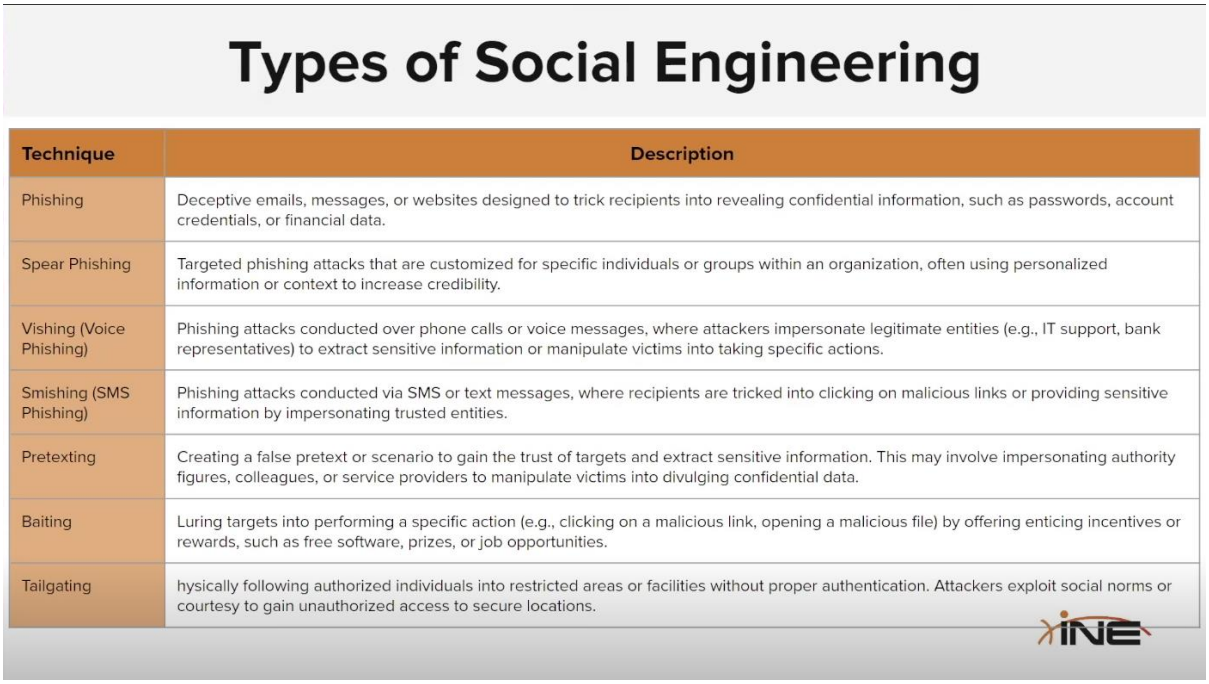
Host & Network Penetration Testing: Social Engineering

Estos son los tipos de ingeniería social que podemos usar.



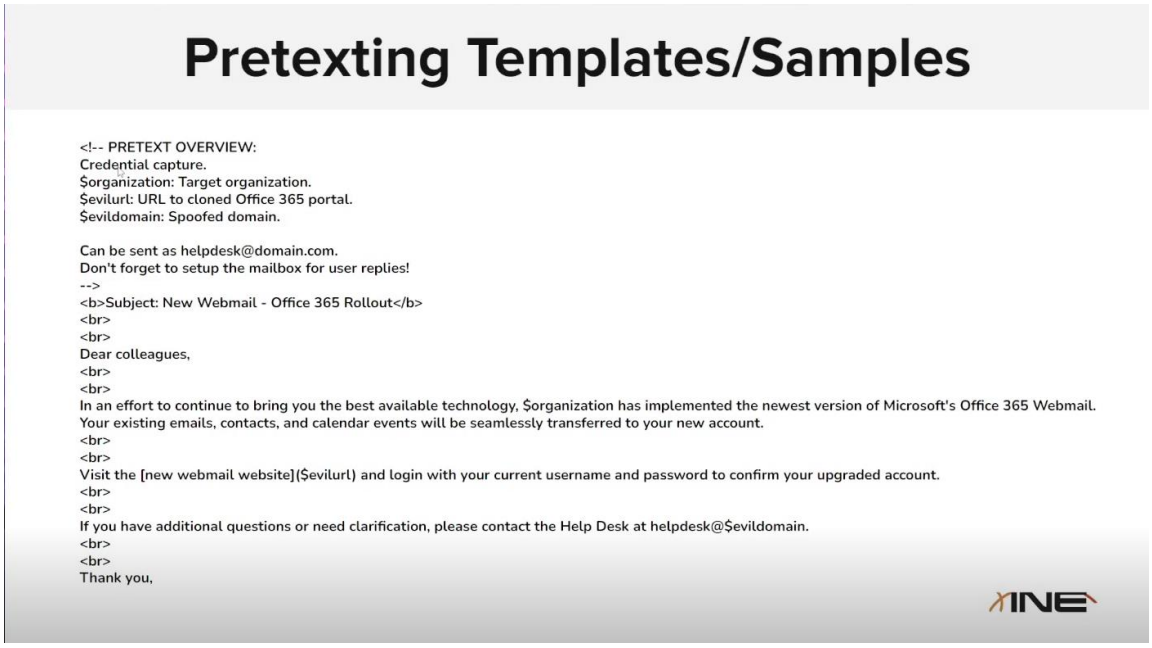
Types of Social Engineering

Technique	Description
Phishing	Deceptive emails, messages, or websites designed to trick recipients into revealing confidential information, such as passwords, account credentials, or financial data.
Spear Phishing	Targeted phishing attacks that are customized for specific individuals or groups within an organization, often using personalized information or context to increase credibility.
Vishing (Voice Phishing)	Phishing attacks conducted over phone calls or voice messages, where attackers impersonate legitimate entities (e.g., IT support, bank representatives) to extract sensitive information or manipulate victims into taking specific actions.
Smishing (SMS Phishing)	Phishing attacks conducted via SMS or text messages, where recipients are tricked into clicking on malicious links or providing sensitive information by impersonating trusted entities.
Pretexting	Creating a false pretext or scenario to gain the trust of targets and extract sensitive information. This may involve impersonating authority figures, colleagues, or service providers to manipulate victims into divulging confidential data.
Baiting	Luring targets into performing a specific action (e.g., clicking on a malicious link, opening a malicious file) by offering enticing incentives or rewards, such as free software, prizes, or job opportunities.
Tailgating	hysically following authorized individuals into restricted areas or facilities without proper authentication. Attackers exploit social norms or courtesy to gain unauthorized access to secure locations.

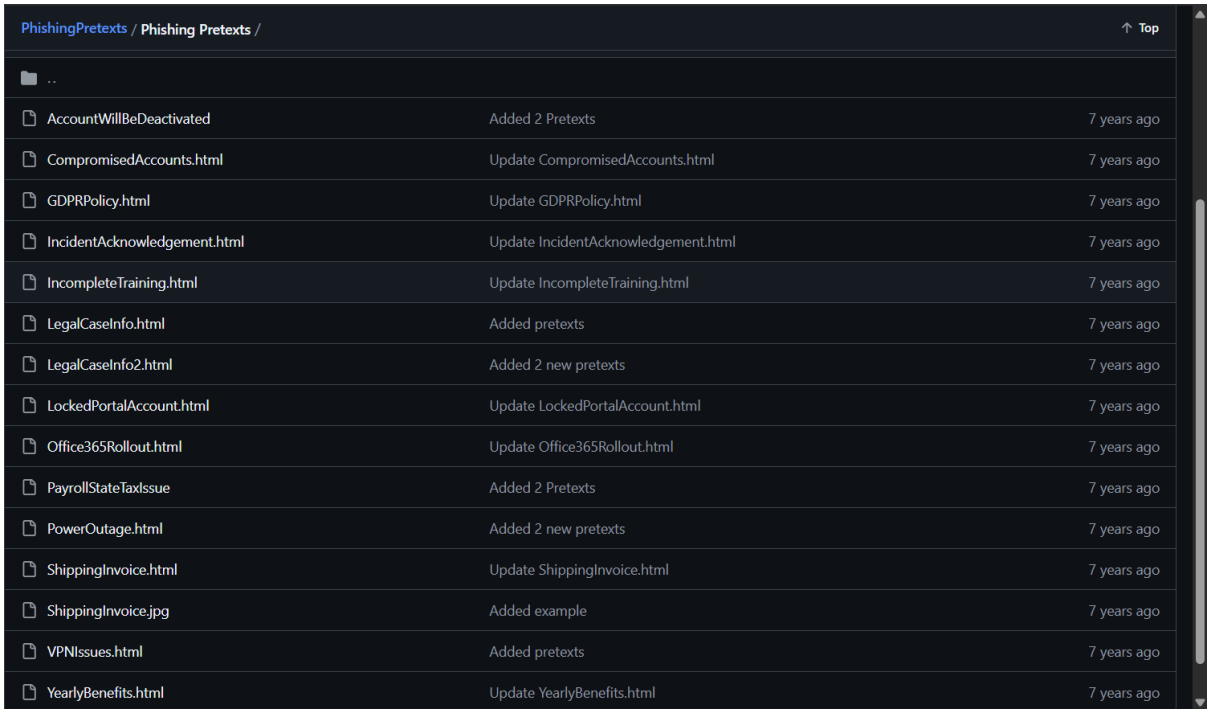
Uno de lo más efectivos es Vishing (Voice Phishing), pero en esta guía nos centraremos el Phishing y Pretexting, también Spear Phishing hasta cierto punto, al menos en una estafa a nivel conceptual.

Pretexting

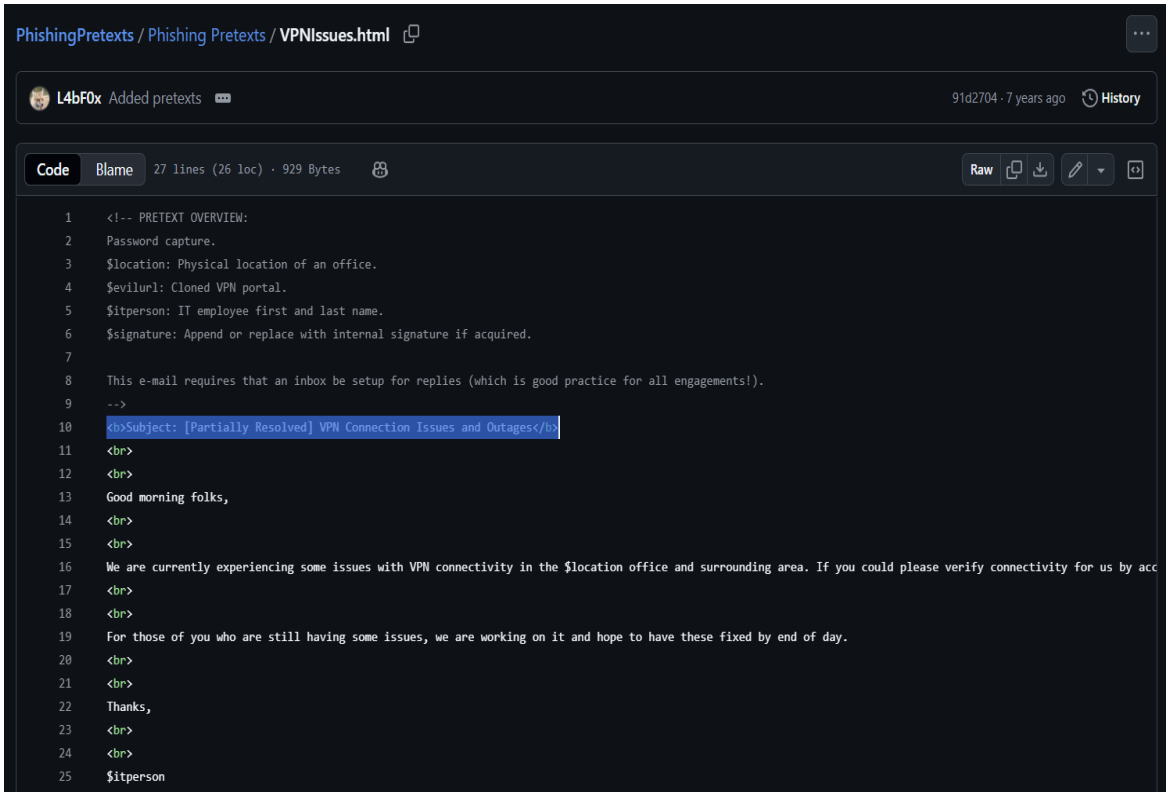
Aquí tenemos una plantilla de Pretexting. Luego veremos donde podemos encontrar estas plantillas ya que nos servirán de utilidad para crear las nuestras.



https://github.com/L4bF0x/PhishingPretexts/tree/master/Phishing%20Pretexts



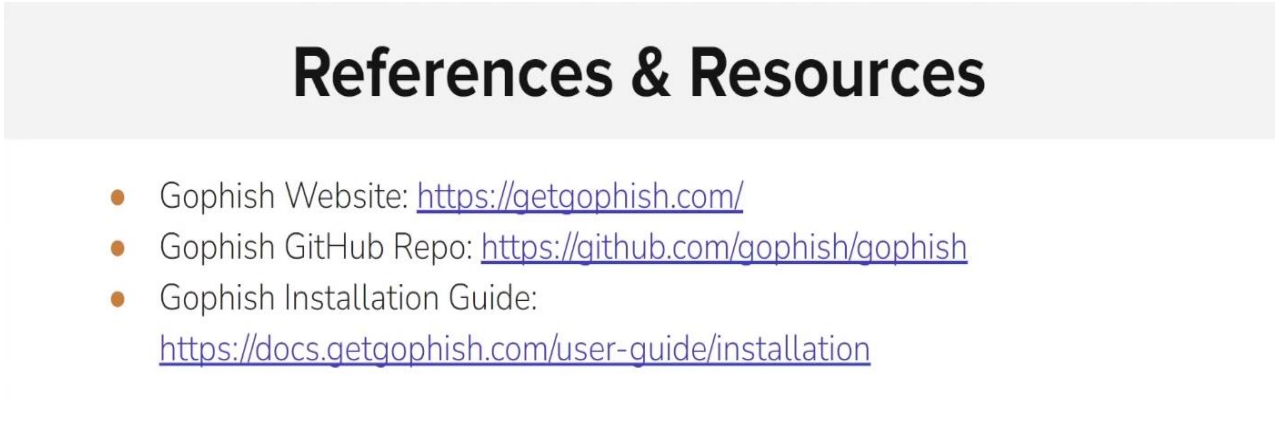
Como podemos ver en la captura de arriba, aparecen diferentes formatos para diferentes objetivos. Por ejemplo, los problemas de VPN son muy comunes en las empresas de tecnología.



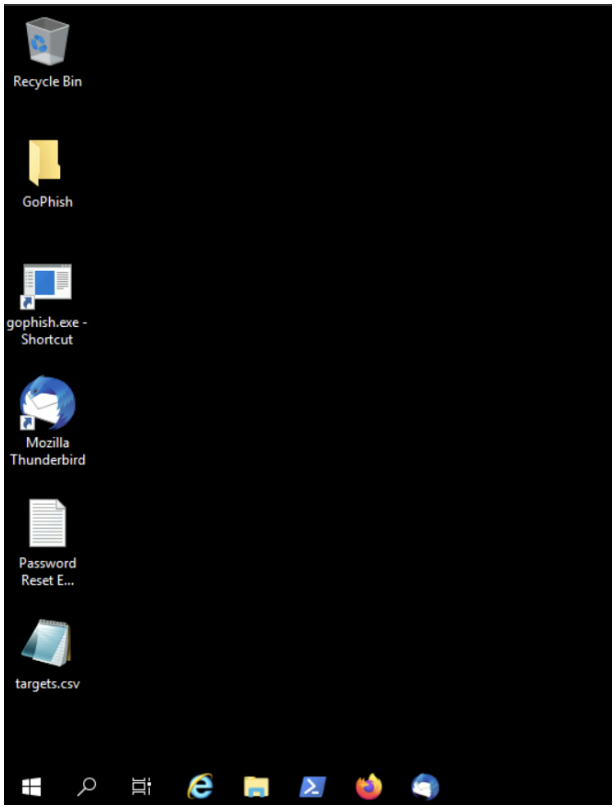
Pero la conclusión es que todos estos ejemplos nos brindarán un excelente punto de partida para que lo utilicemos en compromisos o campañas de pishing.

Phishing with GoPhish – Part 1

Ahora veremos como planificar, establecer y organizar una campaña de Phishing mediante GoPhish.

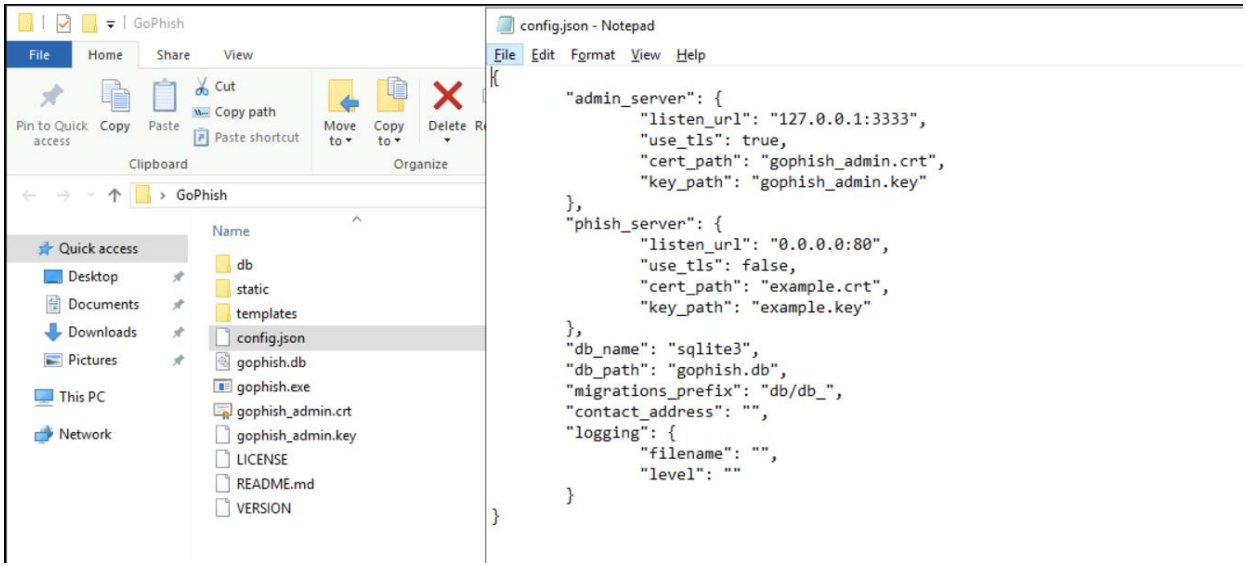


Antes de empezar, vamos a dar un pequeño recorrido de lo que tiene este sistema con respecto a gopshish:



Primero tenemos el repositorio de Gophish que hemos clonado a nuestro sistema, después el ejecutable para que nos sea más cómodo de ejecutarlo directamente desde nuestro escritorio, también tenemos Mozilla Thunderbird que está configurado con un correo electrónico de trabajo de un empleado víctima, la razón por la que tenemos esta configuración es para que podamos ver cómo se ve el correo electrónico phishing cuando se envía a un objetivo o a un empleado, luego tenemos la plantilla que usaremos para el pretexting, en este caso trata sobre el restablecimiento de una contraseña. Y, por último, tenemos una lista de objetivos que, si fuese un pentest real, serían correos que habríamos reunido durante la fase de reconocimiento o identificación de objetivos del cliente.

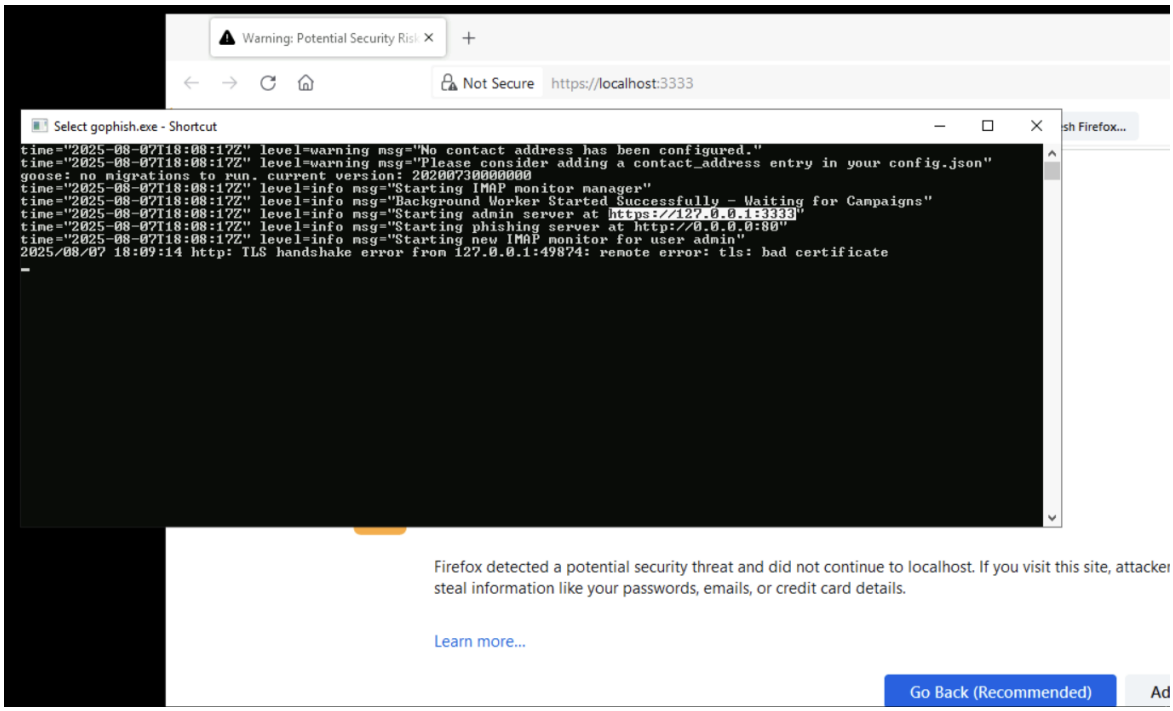
IMPORTANTE: si estamos ante una campaña real para un pentest, tendríamos que configurar Gophish en un servidor Linux en un VPS, puedes elegir el proveedor de tu preferencia y también, probablemente, necesitarás comprar un dominio y luego configurar los correos electrónicos que desees falsificar o los detalles de su correo electrónico del remitente. También vamos a configurar un servidor de correo para enviar correos electrónicos si estas utilizando por ejemplo AWS, podemos utilizar el servicio de correo electrónico simple AWS para realizar el envío de las direcciones de correo electrónico.

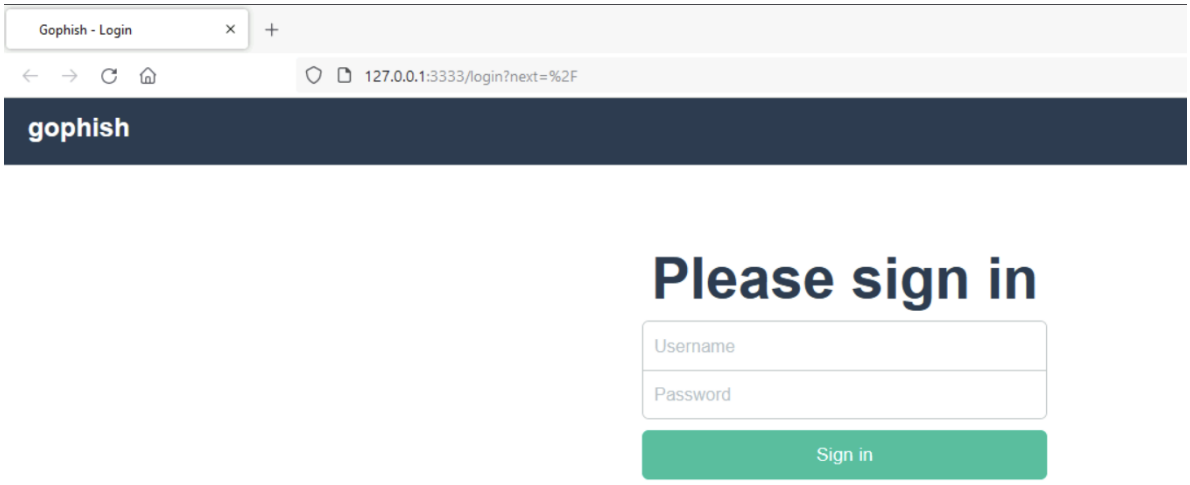


Si estamos usando esto en un pentest real, necesitamos configurar la url a la url de nuestro servidor gophish, si no hemos comprado un dominio, entonces usaremos la dirección IP y en este caso el puerto que el servidor de administración de Gophish ha sido configurado para escuchar en el puerto 3333. También podemos desactivar el tls o ssl si queremos, y luego tenemos el servidor phish que es donde alojaremos una sesión maliciosa y esto está configurado para ejecutarse en el puerto 80.

Bien, una vez comprendido un poco todo esto. Vamos a proceder.

Primero iniciamos el servidor Gophish, lo dicho, si hemos usado nuestro servidor o VPS, lo dejaremos como estaba configurado, en caso de que hayamos comprado un dominio lo cambiaremos, pero no es el caso.

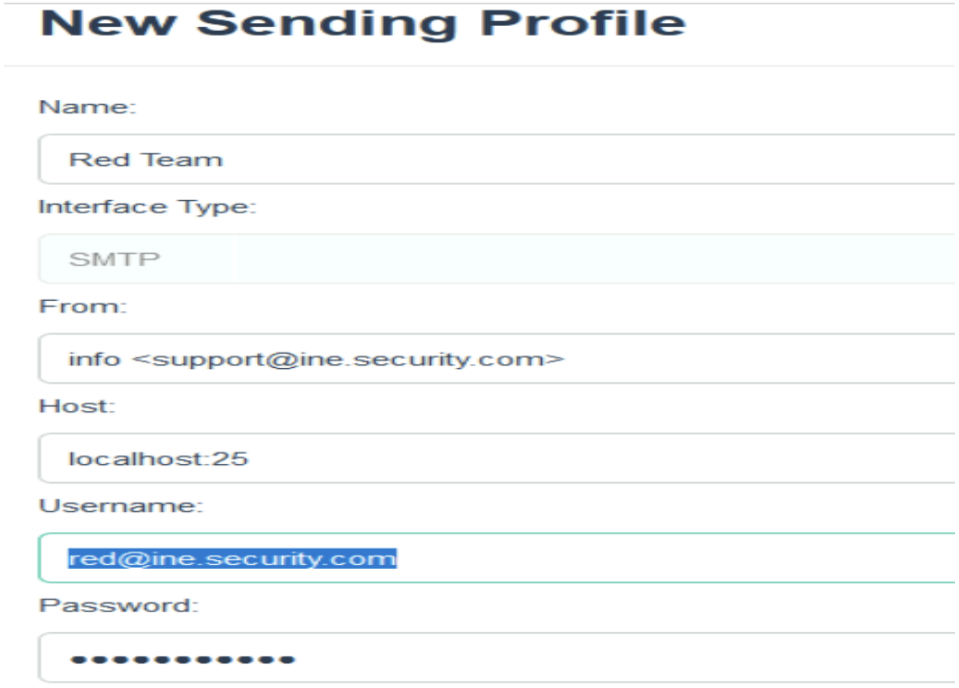




Las credenciales por defecto son admin:phishingpasswd

Ahora lo que vamos a hacer es configurar la campaña. Lo primero que haremos es configurar el perfil de envío.

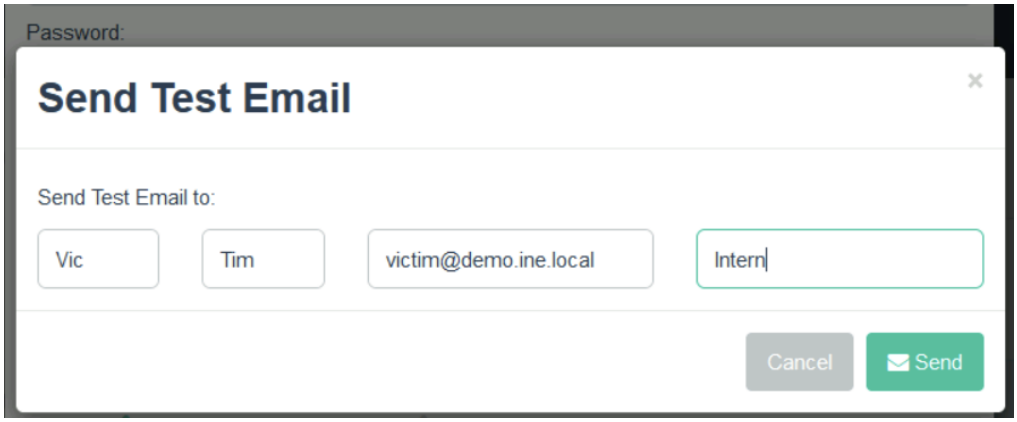
Es lo más importante porque necesitamos un perfil para poder enviar correos electrónicos.

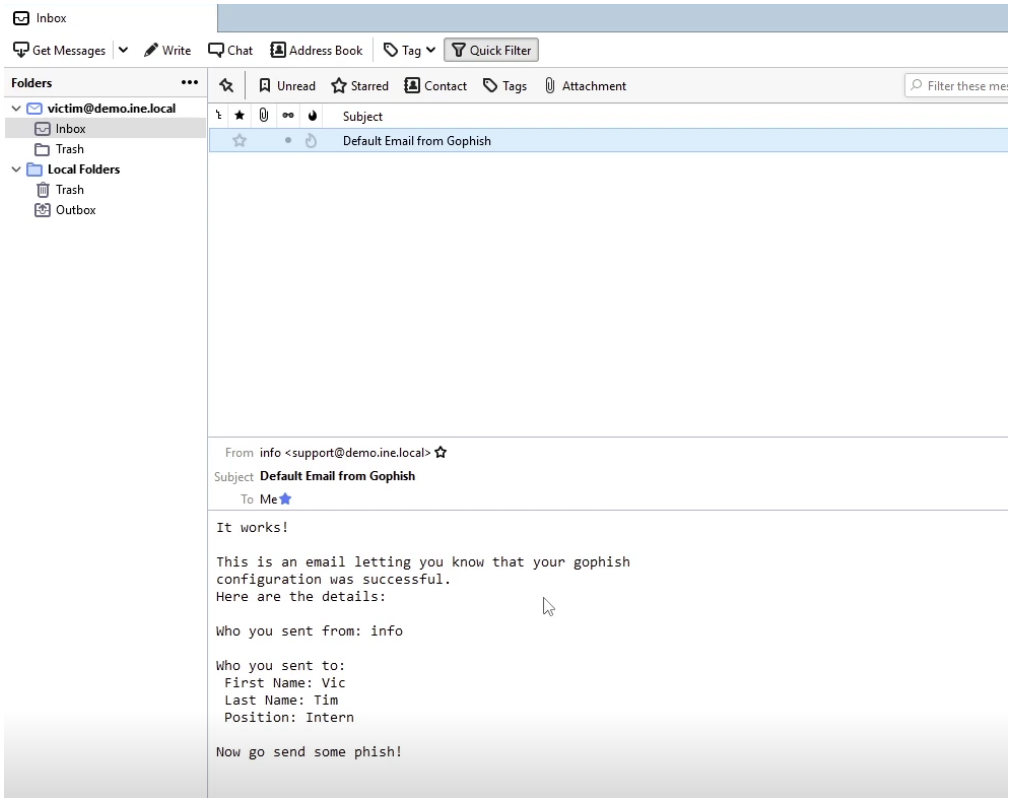


red@ine.security.com será la dirección de correo electrónico del usuario en el servidor de correos y, por supuesto, en un pentest real, tendríamos que especificar nuestro propio dominio o correo electrónico.

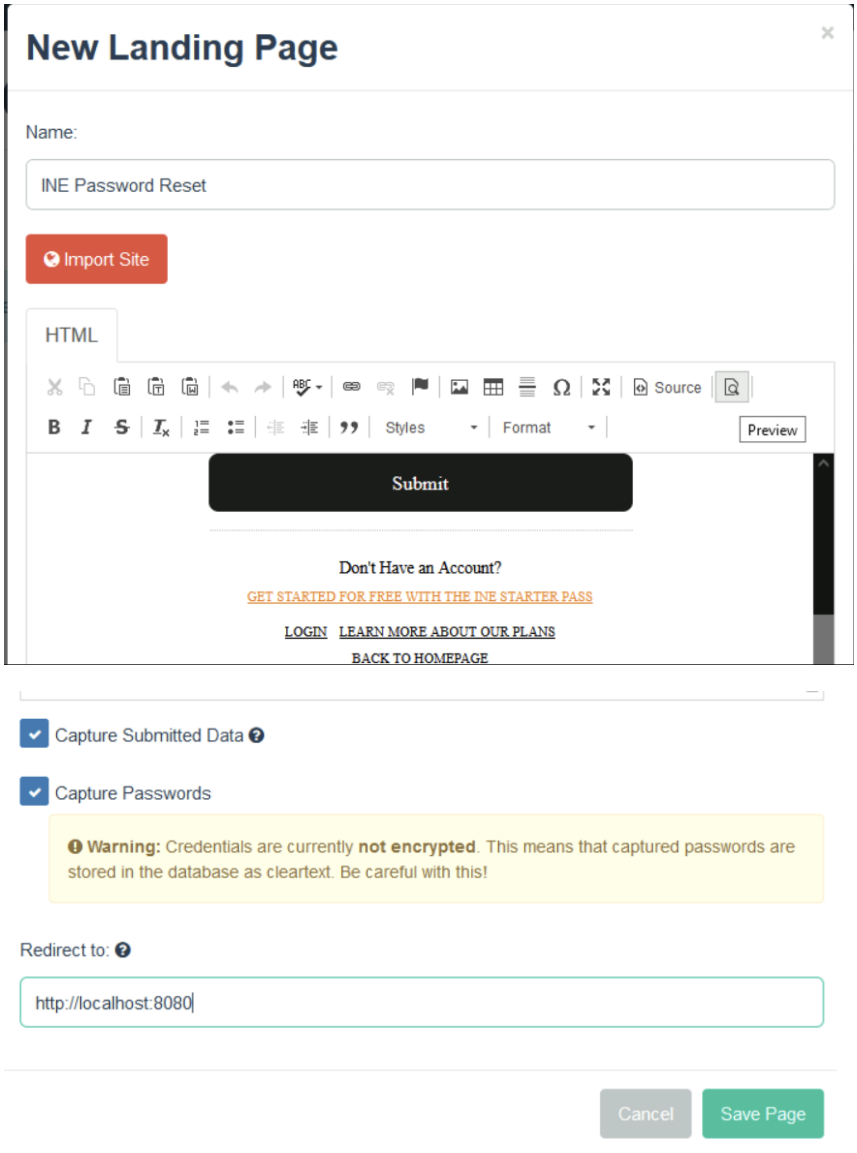
Una vez configurado todo, vamos a hacer un pequeño demo para ver si funciona. Para ello enviaremos un correo a la víctima.

Configuraremos el nombre, apellido, su correo y su posición en la empresa o el cargo que tenga.





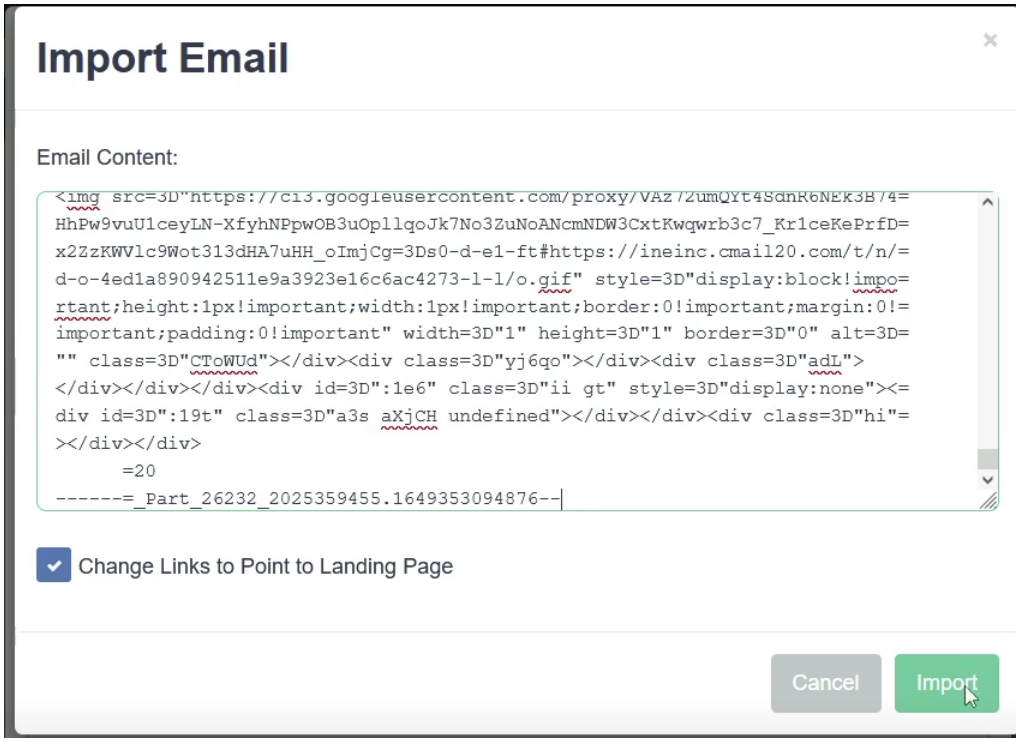
Bien, una vez comprobado esto, vamos a crear una landing page que imitará el portal de inicio de sesión del INE o al menos uno de sus portales de inicio de sesión.

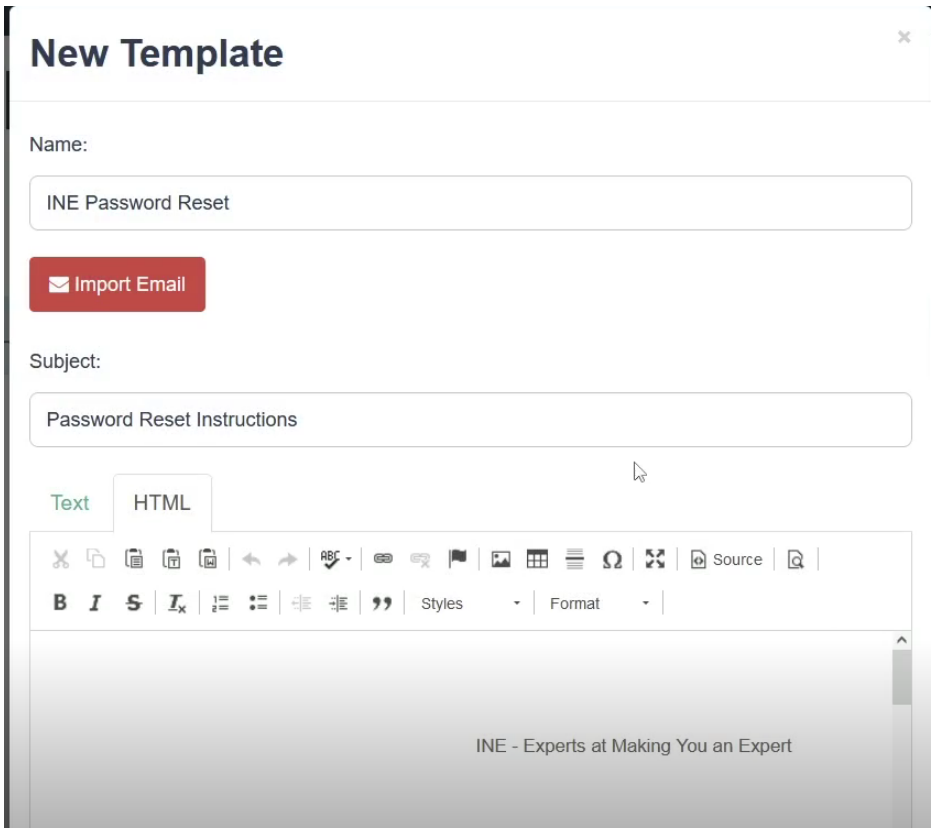


En este caso importaríamos el HTML de la página de inicio de sesión del INE, y la redirección también. Lo hago con localhost debido a que esta máquina no tiene internet y ya tiene descargados el index HTML del INE en ese puerto especifico 8080.

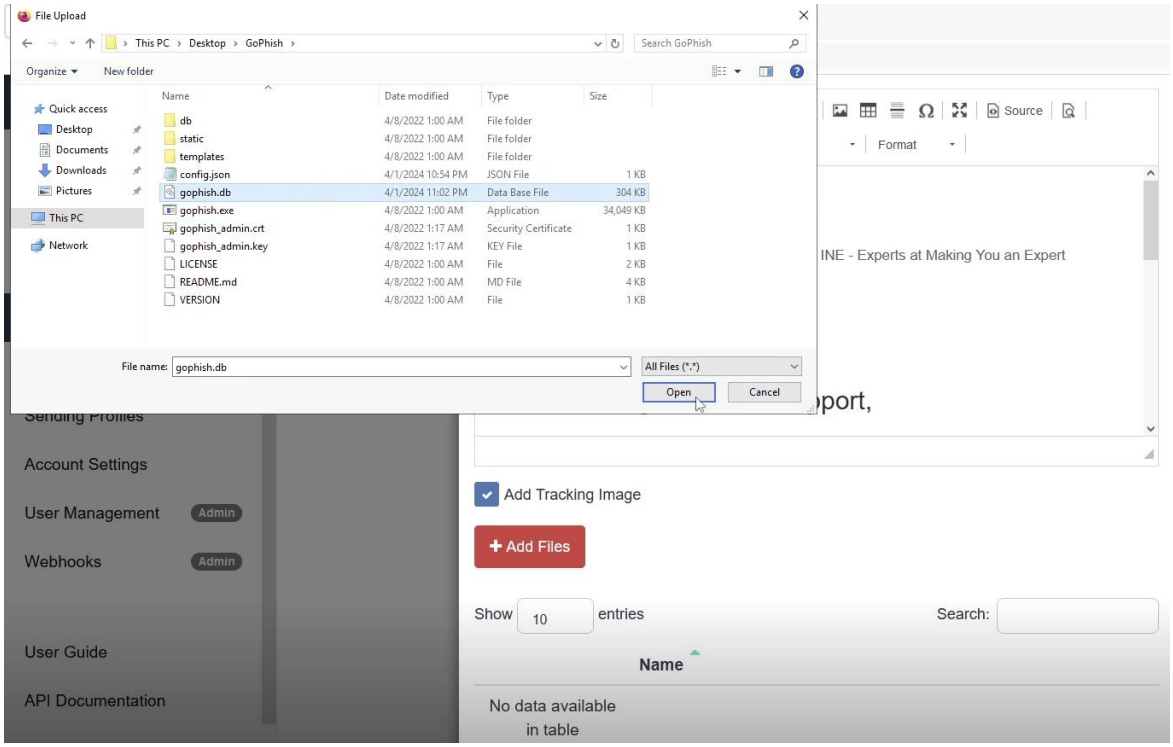
Ahora vamos a crear un email template.

Aquí es donde entra en juego nuestro pretexting.

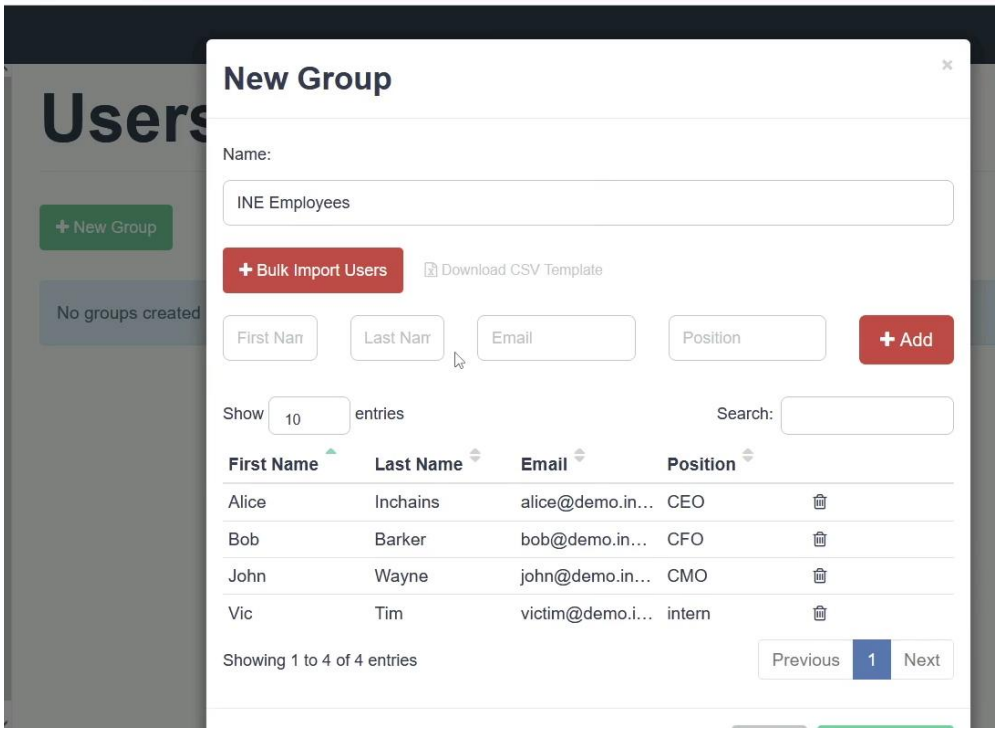




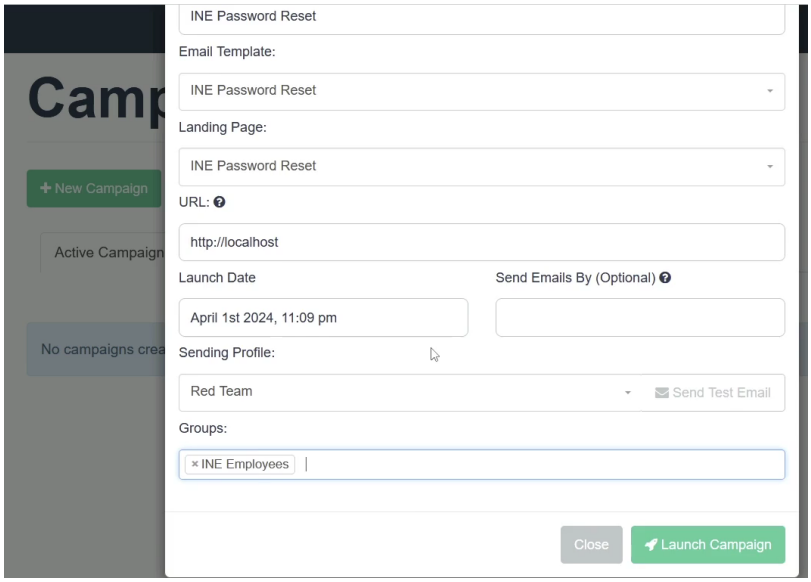
También, si quisieramos añadir algun payload malicioso podríamos añadirlo.

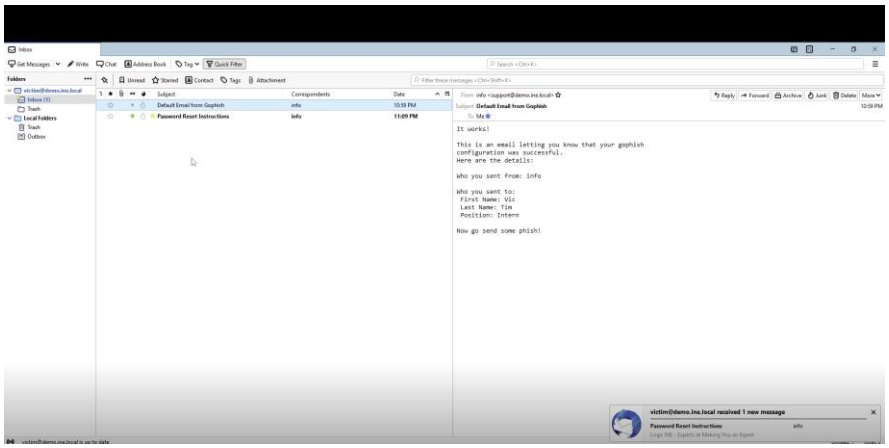


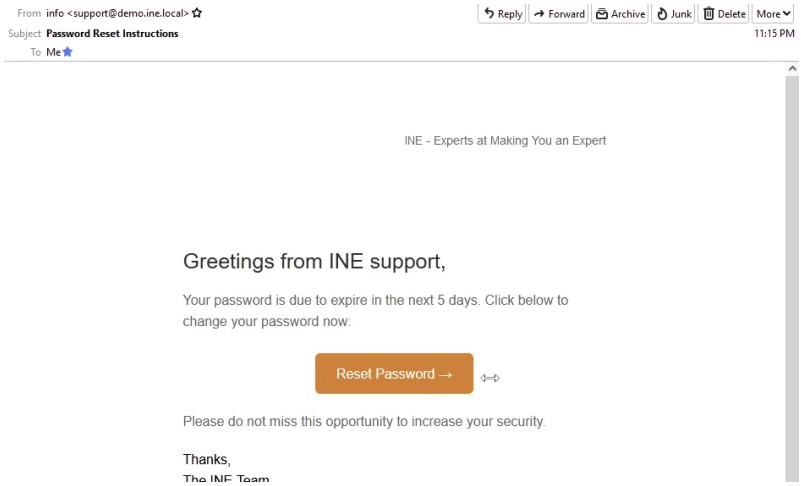
Ahora vamos a crear grupos, primero crearemos un grupo de “empleados” del INE.

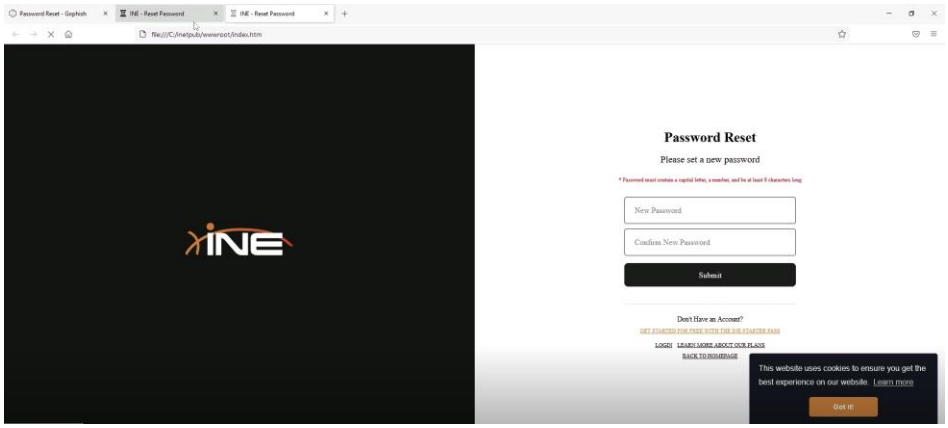


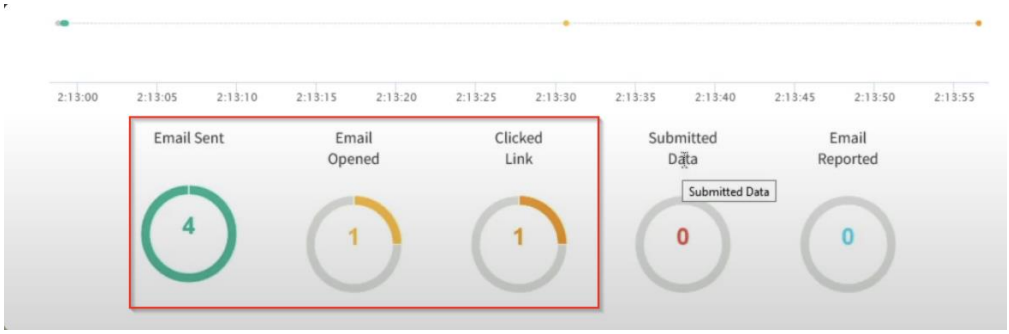
Ahora sí estamos listos para lanzar nuestra campaña.


INE Password Reset
Email Template:
INE Password Reset
Landing Page:
INE Password Reset
URL:
http://localhost
Launch Date
Send Emails By (Optional)
April 1st 2024, 11:09 pm
Sending Profile:
Red Team
Send Test Email
Groups:
INE Employees
Close
Launch Campaign




From info <support@demo.ine.local>
Subject Password Reset Instructions
To Me
Reply
Forward
Archive
Junk
Delete
More
11:15 PM
INE - Experts at Making You an Expert
Greetings from INE support,
Your password is due to expire in the next 5 days. Click below to change your password now:
Reset Password →
Please do not miss this opportunity to increase your security.
Thanks,





Web Application Penetration Testing: Introduction to the Web & HTTP Protocol

Web App Pentesting vs Web App Security Testing

Aspect	Web App Security Testing	Web App Pentesting
Objective	Identify vulnerabilities and weaknesses in the web application without actively exploiting them.	Actively attempt to exploit identified vulnerabilities and assess the organization's response to attacks.
Focus	Broader in scope, includes both manual and automated testing techniques.	Specific to identifying vulnerabilities and exploiting them, mainly a manual process.
Methodology	Various types of assessments, such as SAST, DAST, IAST, SCA, etc.	Manual testing using tools and techniques to simulate real-world attacks.
Exploitation	Does not involve exploitation of vulnerabilities.	Involves controlled exploitation to validate vulnerabilities.
Impact	Non-intrusive; primarily focused on identifying issues.	Can be intrusive, may cause application disruption during testing.
Reporting	Identifies vulnerabilities and provides remediation recommendations.	Documents successful exploits, identifies weaknesses, and recommends remediation measures.
Testing Approach	May include automation for vulnerability scanning.	Primarily manual, using manual testing techniques and tools.
Goal	Enhance overall security posture of the web application.	Validate the effectiveness of existing security controls and incident response capabilities.

Common Web Application Threats & Risks

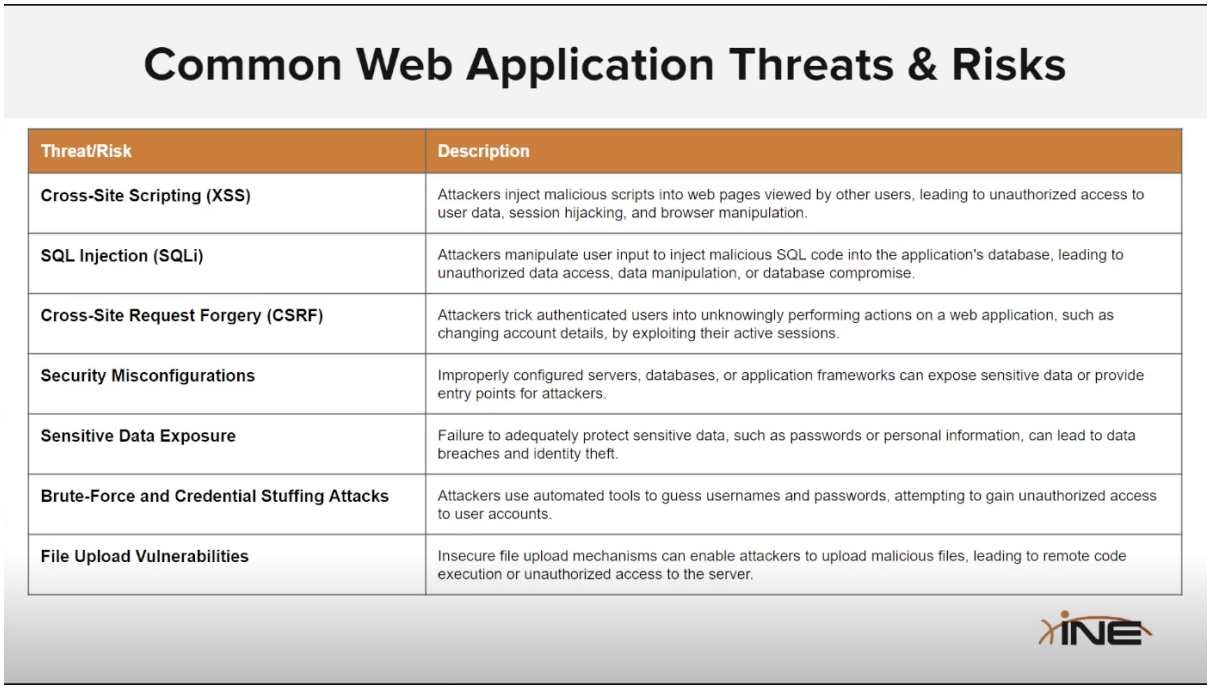
¿Qué es una amenaza?

En el contexto de la ciberseguridad, una amenaza se refiere a cualquier potencial fuente de daño o evento adverso que puede explotar una vulnerabilidad en un sistema o medidas de seguridad de una organización.

Pueden incluir varios tipos de ataques como infección de malwares, intentos de phishing, denegación de ataques a servicios y filtración de datos.

¿Qué es un riesgo?

El riesgo es el potencial de pérdida o daño resultante de una amenaza que explota una vulnerabilidad en un sistema u organización.



Common Web Application Threats & Risks

Threat/Risk	Description
Cross-Site Scripting (XSS)	Attackers inject malicious scripts into web pages viewed by other users, leading to unauthorized access to user data, session hijacking, and browser manipulation.
SQL Injection (SQLi)	Attackers manipulate user input to inject malicious SQL code into the application's database, leading to unauthorized data access, data manipulation, or database compromise.
Cross-Site Request Forgery (CSRF)	Attackers trick authenticated users into unknowingly performing actions on a web application, such as changing account details, by exploiting their active sessions.
Security Misconfigurations	Improperly configured servers, databases, or application frameworks can expose sensitive data or provide entry points for attackers.
Sensitive Data Exposure	Failure to adequately protect sensitive data, such as passwords or personal information, can lead to data breaches and identity theft.
Brute-Force and Credential Stuffing Attacks	Attackers use automated tools to guess usernames and passwords, attempting to gain unauthorized access to user accounts.
File Upload Vulnerabilities	Insecure file upload mechanisms can enable attackers to upload malicious files, leading to remote code execution or unauthorized access to the server.

Common Web Application Threats & Risks

Threat/Risk	Description
Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS)	DoS and DDoS attacks aim to overwhelm web application servers, causing service disruptions and denying legitimate users access.
Server-Side Request Forgery (SSRF)	Attackers use SSRF to make requests from the server to internal resources or external networks, potentially leading to data theft or unauthorized access.
Inadequate Access Controls	Weak access controls may allow unauthorized users to access restricted functionalities or sensitive data.
Using Components with Known Vulnerabilities	Integrating third-party components with known security flaws can introduce weaknesses into the web application.
Broken Access Control	Inadequate access controls can allow unauthorized users to access restricted functionalities or sensitive data.

Web Application Architecture (IMPORTANTE)

Client-Server Model

Las aplicaciones web normalmente se crean en el modelo del servidor cliente. En este tipo de arquitectura, la aplicación web se divide en dos componentes principales. Cliente y servidor.

El cliente representa la interfaz de usuario y la interacción del usuario con la aplicación web. Es la interfaz del acceso web a nuestros navegadores web. El cliente es responsable de mostrar las páginas web, manejar la entrada del usuario y envío de solicitudes al servidor de datos o acciones.

El servidor representa el back-end de la aplicación web. Procesa las solicitudes de los clientes, ejecuta la lógica empresarial de las aplicaciones, se comunica con la base de datos y otros servicios, y genera respuestas y lo envía de vuelta al cliente.

Web Application Components

Component	Function
User Interface (UI)	The user interface is the visual presentation of the web application seen and interacted with by users. It includes elements such as web pages, forms, menus, buttons, and other interactive components.
Client-Side Technologies	Client-side technologies, such as HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript, are used to create the user interface and handle interactions directly within the user's web browser.
Server-Side Technologies	Server-side technologies, such as programming languages (e.g., PHP, Python, Java, Ruby) and frameworks, are used to implement the application's business logic, process requests from clients, access databases, and generate dynamic content to be sent back to the client.
Databases	Databases are used to store and manage the web application's data. They store user information, content, configurations, and other relevant data required for the application's operation.
Application Logic	The application logic represents the rules and procedures that govern how the web application functions. It includes user authentication, data validation, security checks, and other business rules.
Web Servers	Web servers handle the initial request from clients and serve the client-side components, such as static files (HTML, CSS, JavaScript), to the users.
Application Servers	Application servers execute the server-side code and handle the dynamic processing of client requests. They communicate with databases, perform business logic, and generate dynamic content.

Client-side Processing

El procesamiento del lado del cliente implica la ejecución de tareas y cálculos en el dispositivo del usuario, normalmente dentro de nuestro navegador web. No es adecuado para manejar operaciones sensibles o críticas, ya que puede ser fácilmente manipulado por usuarios o actores malintencionados (XSS).

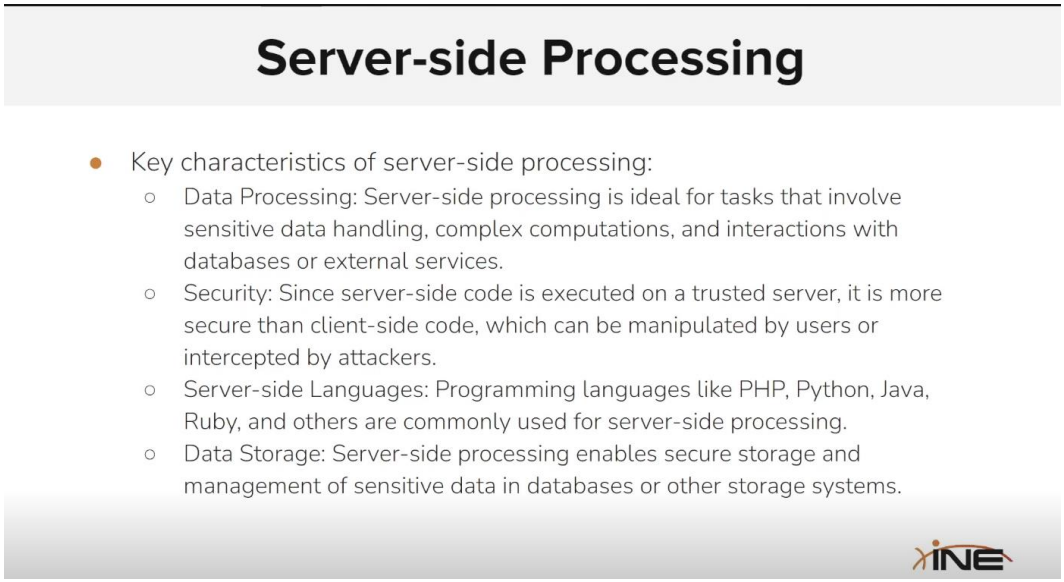
Client-side Processing

- Key characteristics of client-side processing:
  - User Interaction: Client-side processing is well-suited for tasks that require immediate user interaction and feedback, as there is no need to send data back and forth to the server.
  - Responsive User Experience: Since processing happens locally, client-side operations can provide a smoother and more responsive user experience.
  - JavaScript: JavaScript is the primary programming language used for client-side processing. It allows developers to manipulate the web page's content, handle user interactions, and perform validations without involving the server.
  - Data Validation: Client-side validation ensures that user input meets specific criteria before it is sent to the server, reducing the need to make unnecessary server requests.

Server-side Processing

El procesamiento del lado del servidor implica ejecutar tareas y cálculos en el servidor web, que es la computadora remota donde la aplicación web está alojada.

Se refiere al back-end de la aplicación web, donde tiene lugar la lógica empresarial y el procesamiento de datos.

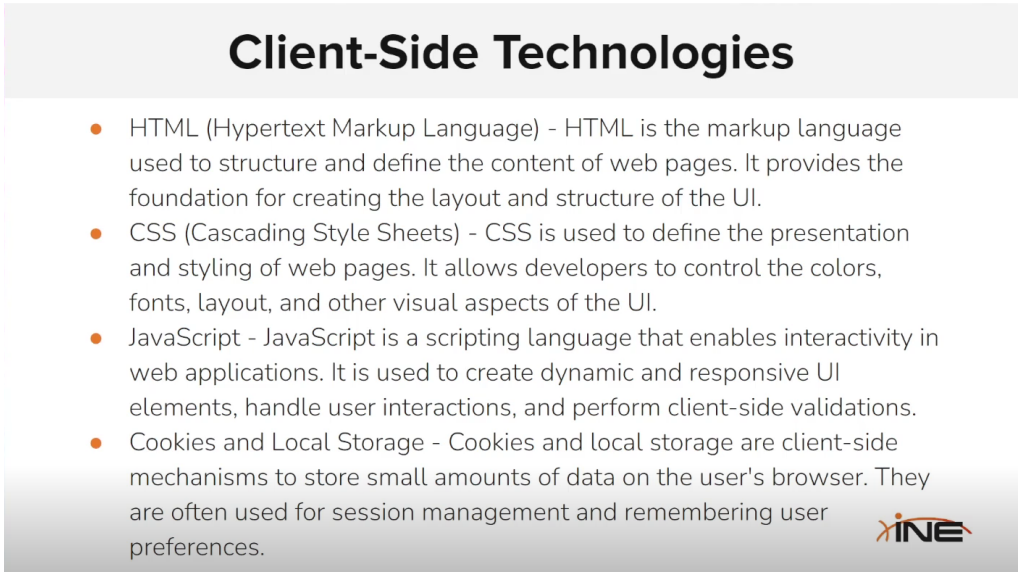


Communication & Data Flow

¿Cómo se comunican las aplicaciones web?

Las aplicaciones web se comunican a través de internet mediante HTTP o HTTPS. Y cuando un usuario interactúa con la aplicación web haciendo click en enlaces o al enviar formularios, el cliente o el navegador envía solicitudes HTTP a el servidor. El servidor procesa estas solicitudes, interactúa con la base de datos, si es necesario, realiza las acciones requeridas y genera una respuesta HTTP, generalmente con el front-end de la página web o cualquier información como resultado de lo que envió al servidor para el procesamiento del lado del servidor desde el archivo.

Web Application Technologies



Server-Side Technologies

- Web Server - The web server is responsible for receiving and responding to HTTP requests from clients (web browsers). It hosts the web application's files, processes requests, and sends responses back to clients. (Apache2, Nginx, Microsoft IIS etc)
- Application Server - The application server runs the business logic of the web application. It processes user requests, accesses databases, and performs computations to generate dynamic content that the web server can serve to clients.
- Database Server - The database server stores and manages the web application's data. It stores user information, content, configurations, and other relevant data required for the application's operation. (MySQL, PostgreSQL, MSSQL, Oracle etc)

Server-Side Technologies

- Server-side Scripting Languages - Server-side scripting languages (e.g., PHP, Python, Java, Ruby) are used to handle server-side processing. They interact with databases, perform validations, and generate dynamic content before sending it to the client.

Communication & Data Flow

- Web applications communicate over the internet using HTTP (Hypertext Transfer Protocol).
- When a user interacts with the web application by clicking on links or submitting forms, the client sends HTTP requests to the server.
- The server processes these requests, interacts with the database if necessary, performs the required actions, and generates an HTTP response.
- The response is then sent back to the client, which renders the content and presents it to the user.

Data Interchange

- Data interchange refers to the process of exchanging data between different computer systems or applications, allowing them to communicate and share information.
- It is a fundamental aspect of modern computing, enabling interoperability and data sharing between diverse systems, platforms, and technologies.
- Data interchange involves the conversion of data from one format to another, making it compatible with the receiving system.
- This ensures that data can be interpreted and utilized correctly by the recipient, regardless of the differences in their data structures, programming languages, or operating systems.

Data Interchange Technologies

- APIs (Application Programming Interfaces) - APIs allow different software systems to interact and exchange data. Web applications use APIs to integrate with external services, share data, and provide functionalities to other applications.

Data Interchange Protocols

- JSON (JavaScript Object Notation) - JSON is a lightweight and widely used data interchange format that is easy for both humans and machines to read and write. It is based on JavaScript syntax and primarily used for transmitting data between a server and a web application as an alternative to XML.
- XML (eXtensible Markup Language) - XML is a versatile data interchange format that uses tags to define the structure of the data. It allows users to create their custom tags and define complex hierarchical data structures. XML is commonly used for configuration files, web services, and data exchange between different systems.

Data Interchange Protocols

- REST (Representational State Transfer) - REST is a software architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE) for data interchange. It is widely used for creating web APIs that allow applications to interact and exchange data over the internet.
- SOAP (Simple Object Access Protocol) - SOAP is a protocol for exchanging structured information in the implementation of web services. It uses XML as the data interchange format and provides a standardized method for communication between different systems.

Security Technologies

- Authentication and Authorization Mechanisms - Authentication verifies the identity of users, while authorization controls access to different parts of the web application based on user roles and permissions.
- Encryption (SSL/TLS) - SSL (Secure Socket Layer) or TLS (Transport Layer Security) is used to encrypt data transmitted between the client and server, ensuring secure communication and data protection.

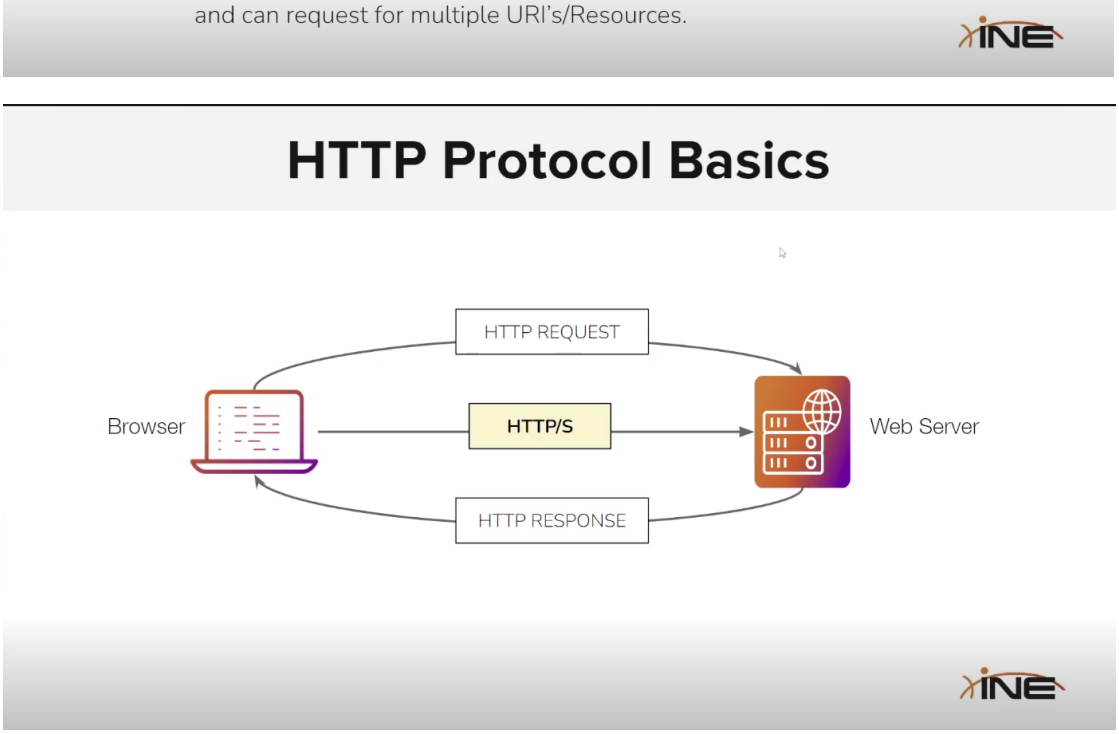
External Technologies

- Content Delivery Networks (CDNs) - CDNs are used to distribute static content (e.g., images, CSS files, JavaScript libraries) to multiple servers located worldwide, improving the web application's performance and reliability.
- Third-Party Libraries and Frameworks - Web applications often leverage third-party libraries and frameworks to speed up development and access advanced features.

HTTPS Protocol Fundamentals (IMPORTANTE)

HTTP Protocol

- HTTP (Hypertext Transfer Protocol) is a stateless application layer protocol used for the transmission of resources like web application data and runs on top of TCP.
- It was specifically designed for communication between web browsers and web servers.
- HTTP utilizes the typical client-server architecture for communication, whereby the browser is the client, and the web server is the server.
- Resources are uniquely identified with a URL/URI.
- HTTP has 2 versions; HTTP 1.0 & HTTP 1.1.
  - HTTP 1.1 is the most widely used version of HTTP and has several advantages over HTTP 1.0 such as the ability to re-use the same connection and can request for multiple URI's/Resources.



HTTP Request – Part 1 (MUY IMPORTANTE)

Request line (línea de solicitud) suele ser la primera línea de cualquier solicitud HTTP y contiene los tres siguientes componentes:

HTTP method se refiere a GET, POST, PUT, DELETE, etc: indica el tipo solicitud que se está realizando.
La URL (Uniform Resource Locator): es la dirección del recurso al que el cliente desea acceder.

La versión HTTP es el protocolo que está siendo usado, por ejemplo, HTTP 1.1

Request Headers en el contexto de HTTP proporcionan información adicional sobre las requests (solicitudes) del servidor web. ¿Cuáles son algunos de los más comunes que veremos?

User-Agent: Contiene información sobre el cliente que realiza las solicitudes

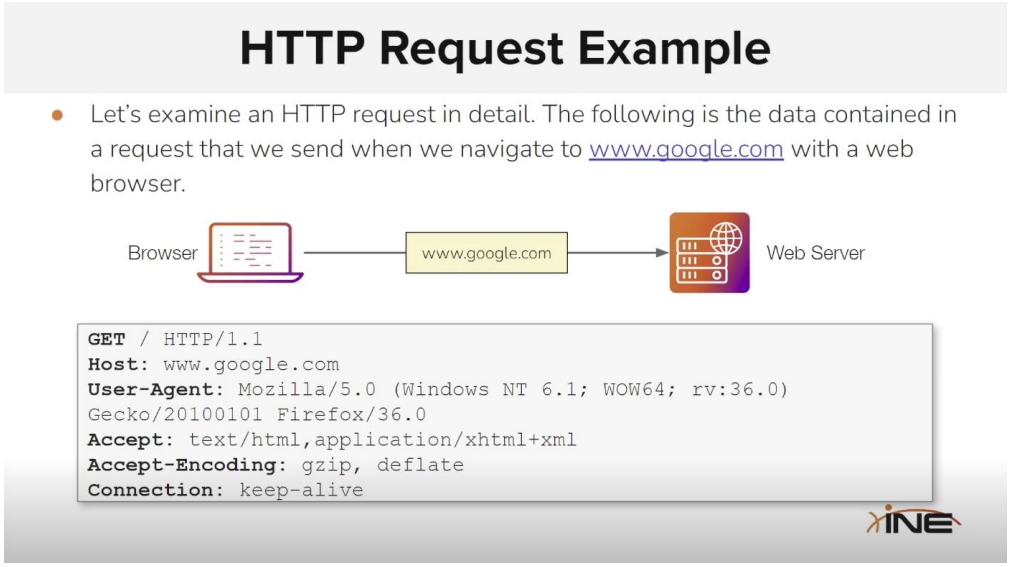
Host: donde se aloja el nombre del servidor, o mejor dicho el nombre del servidor.

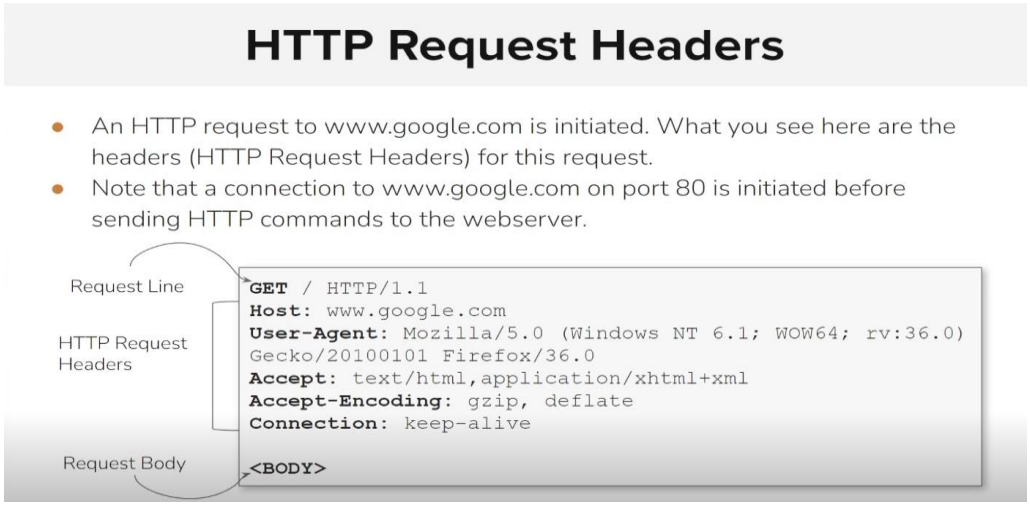
Accept: Los tipos de medios que el cliente puede gestionar en respuesta.

Authorization: Credenciales para autenticación, si se solicita.

Cookie: información almacenada en el lado del cliente y enviado de vuelta al servidor con cada solicitud.

Request body (opcional): algunos métodos HTTP como POST o PUT incluyen un cuerpo de solicitud donde la información se envía al servidor, normalmente en JSON u otro formato de información.

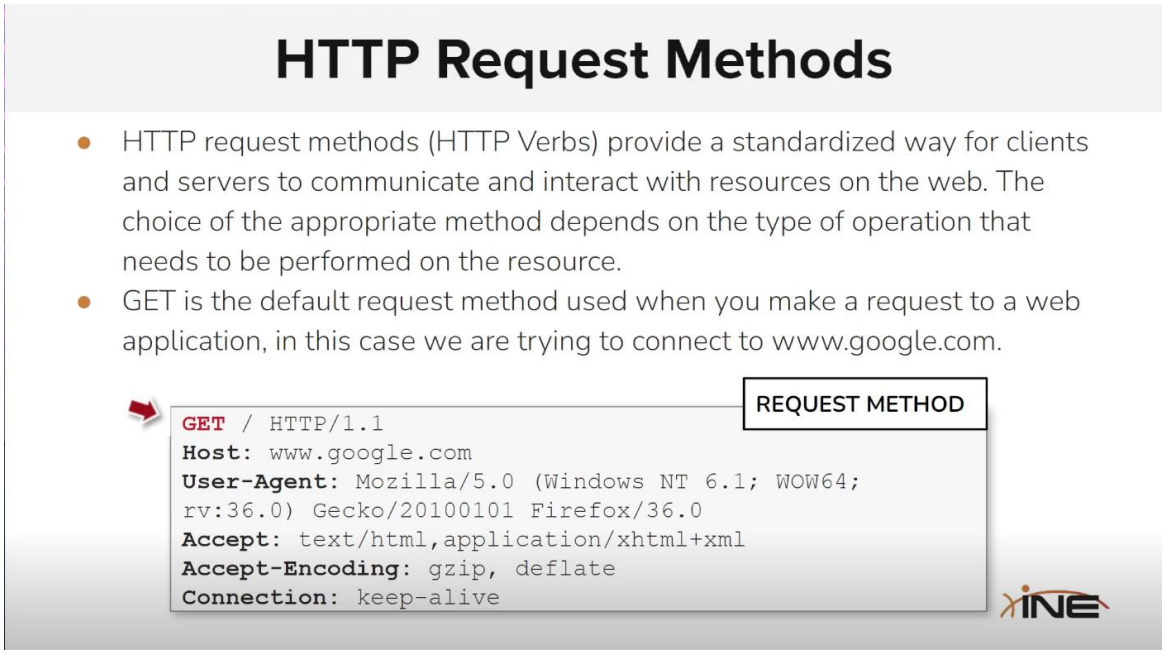


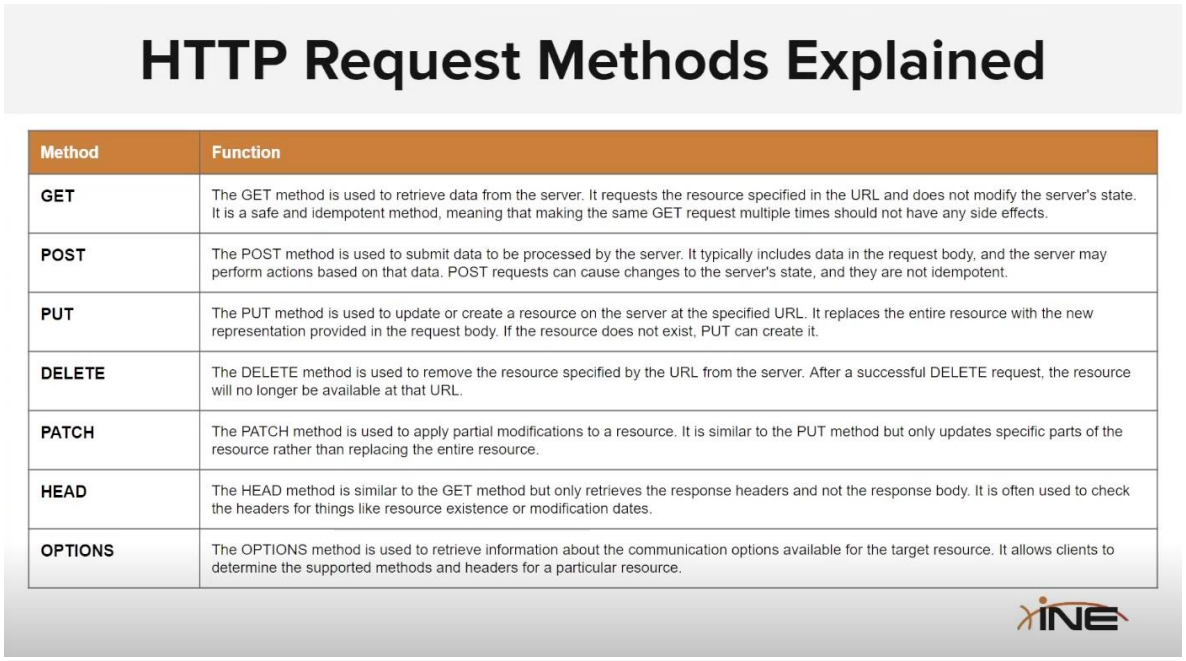


HTTP Request – Part 2 (MUY IMPORTANTE)

HTTP Request Methods proporcionan una forma estandarizada para clientes y web. La elección del método apropiado depende del tipo de operación que debe realizarse en el recurso.

GET es el método por defecto usado cuando haces una solicitud a la aplicación web.





HTTP Response (MUY IMPORTANTE)

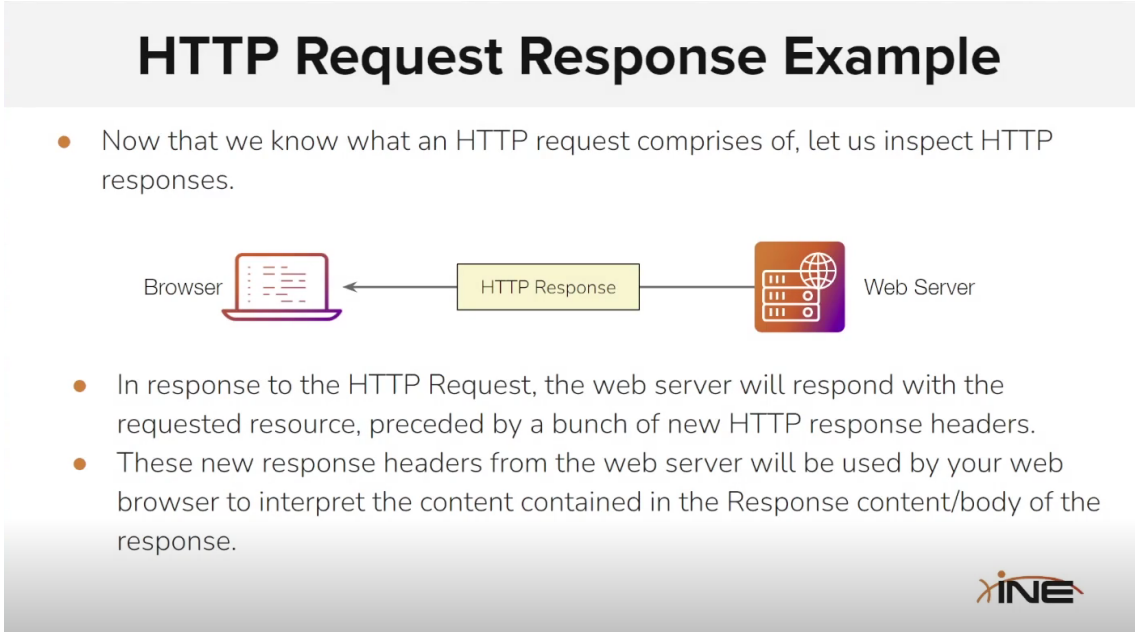
Response Headers son muy similares a los encabezados de solicitud, proporcionan información adicional sobre la respuesta. Los encabezados comunes incuyen:

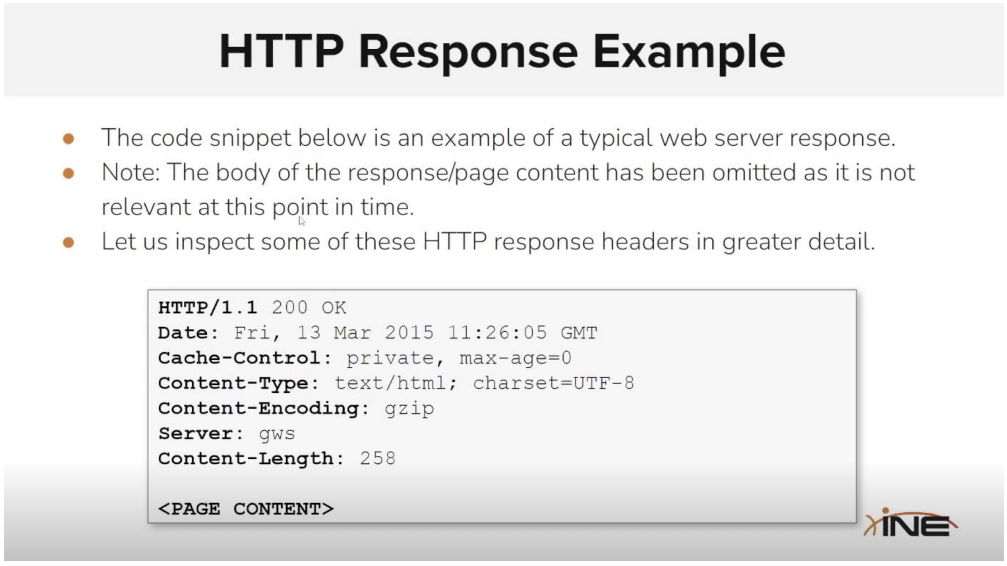
Content-Type: Esto se refiere al tipo o al formato del contenido de la respuesta.

Content-Length: tamaño del response body en bytes.

Set-cookie: Usado para asignar cookies en el lado del cliente para secuencias de respuestas. Por ejemplo, una cookie vinculada a una autenticación a la cual no estás conectado. Así que la próxima vez que intentes hacer una solicitud o lo intentes no se cerrará la sesión.

Cache-control: especifica directivas o reglas para el comportamiento de almacenamiento caché.





Common HTTP Status Codes

Status Code	Meaning
200 OK	The request was successful, and the server has returned the requested data.
301 Moved Permanently	The requested resource has been permanently moved to a new URL, and the client should use the new URL for all future requests.
302 Found	The requested resource is temporarily located at a different URL. This code is commonly used for temporary redirections, but it's often better to use 303 or 307 instead.
400 Bad Request	The server cannot process the request due to a client error (e.g., malformed request syntax).
401 Unauthorized	Authentication is required, and the client must provide valid credentials to access the requested resource.
403 Forbidden	The server understood the request, but the client does not have permission to access the requested resource.
404 Not Found	The requested resource could not be found on the server.
500 Internal Server Error	The server encountered an error while processing the request, and the specific cause is not provided.

Cache-Control Directives

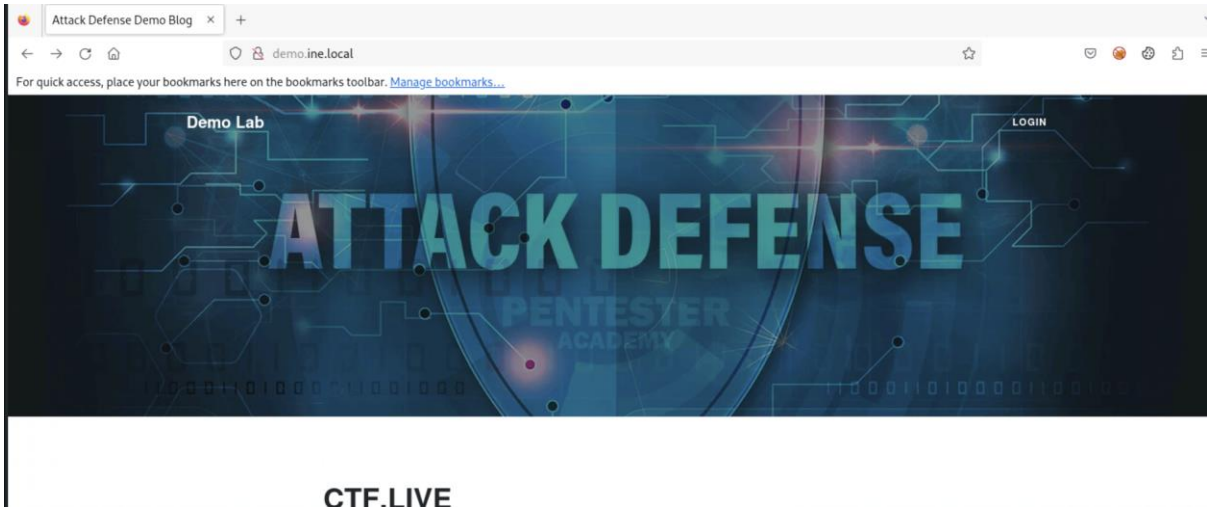
Directive	Meaning
Public	Indicates that the response can be cached by any intermediary caches (such as proxy servers) and shared across different users.
Private	Specifies that the response is intended for a specific user and should not be cached by intermediate caches.
no-cache	Instructs the client to revalidate the response with the server before using the cached version. It does not prevent caching but requires revalidation.
no-store	Directs the client and intermediate caches not to store any version of the response. It ensures that the response is not cached in any form.
max-age=<SECONDS>	Specifies the maximum amount of time in seconds that the response can be cached by the client. After this period, the client should revalidate with the server.

Ahora pasemos a la parte práctica.

Primero de todo realizaremos un escaneo de puertos abiertos.

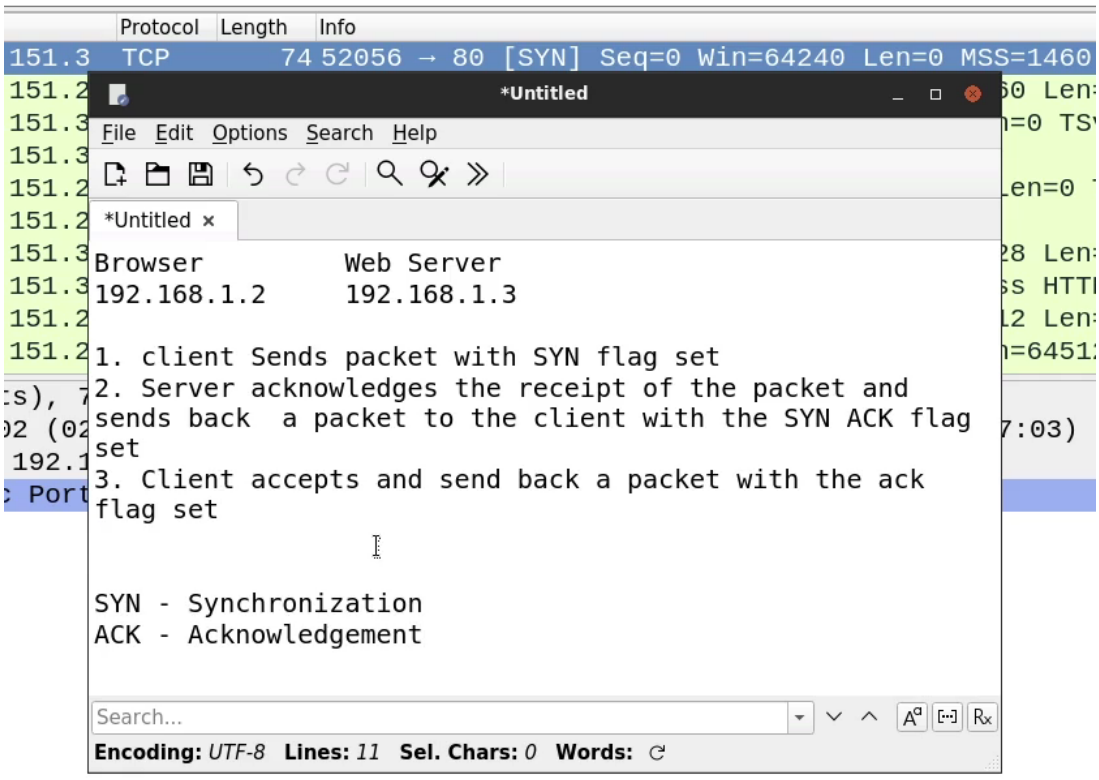
```
Nmap scan report for demo.ine.local (192.17.248.3)
Host is up, received arp-response (0.000021s latency).
Scanned at 2025-08-09 05:51:52 IST for 14s
Not shown: 65533 closed tcp ports (reset)
PORT    STATE SERVICE  REASON         VERSION
80/tcp  open  http     syn-ack ttl 64 Apache httpd (PHP 5.5.9-1ubuntu4.25)
443/tcp open  ssl/http syn-ack ttl 64 Apache httpd
MAC Address: 02:42:C0:11:F8:03 (Unknown)
```

Tenemos el puerto 80 abierto, y podemos ver que tenemos corriendo un servidor apache.



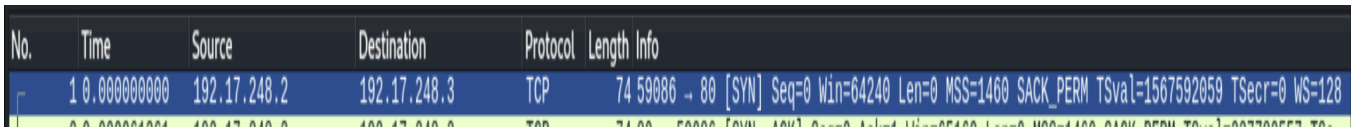
Bien, una vez localizado nuestro primer vector, vamos a pasar a la primera fase o el primer paso que tenemos que analizar antes de llegar a la modificación real de las solicitudes.

Vamos a echar un vistazo en la comunicación, pero también en la comunicación TCP. Y la mejor manera de hacerlo es mediante Wireshark.

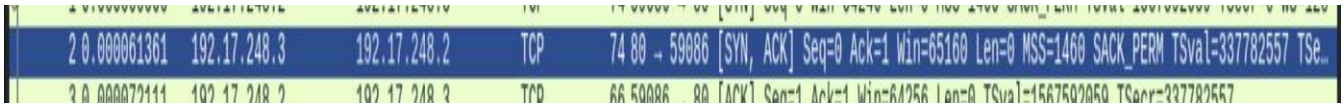


Explicación:

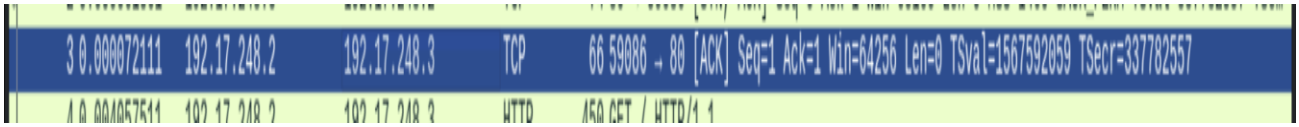
Podemos ver que cuando cargamos la página web. Tenía un paquete TCP de comunicación que iba desde nuestro sistema a la dirección IP del servidor web. Y se estaba conectando al puerto 80 en la dirección IP del servidor web. Y podemos ver que este paquete tenía configurado la flag SYN (mirar captura de arriba).



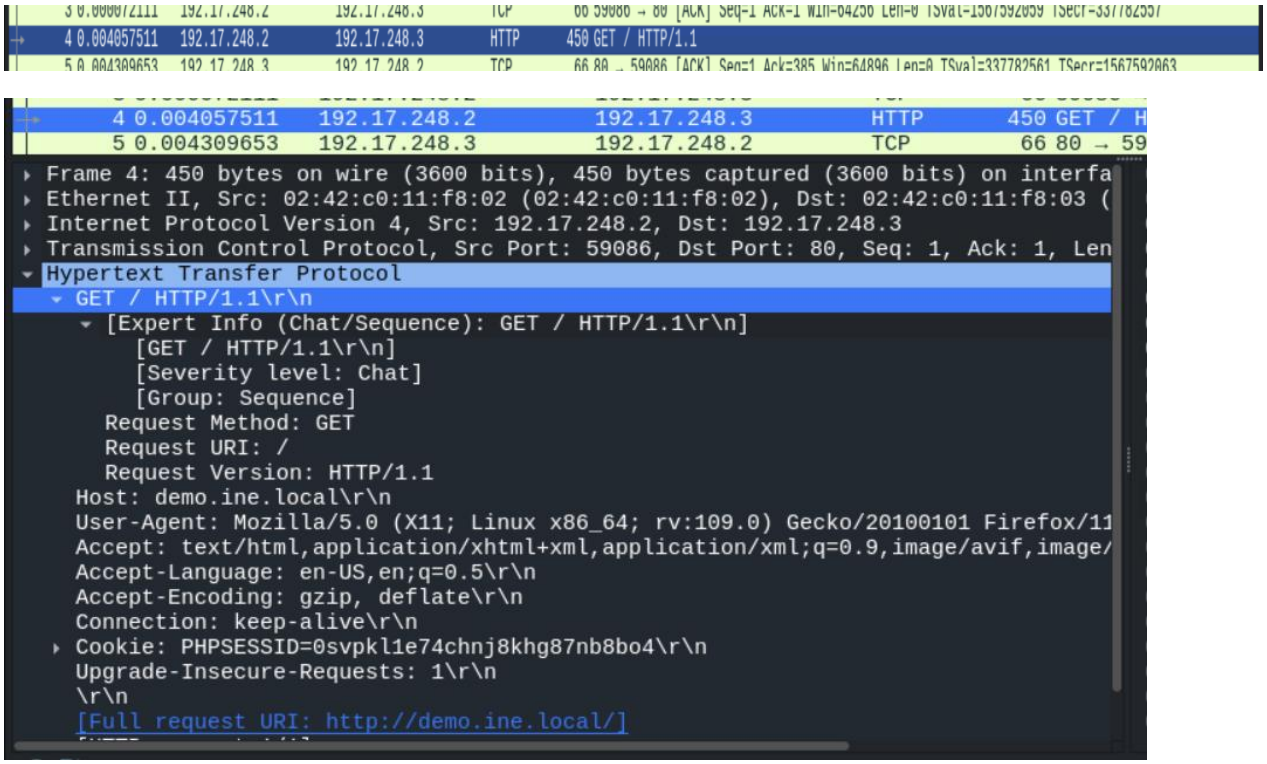
Luego vimos que la dirección IP del servidor web nos respondió a nuestra dirección IP. Y respondió com un paquete flag SYN ACK.



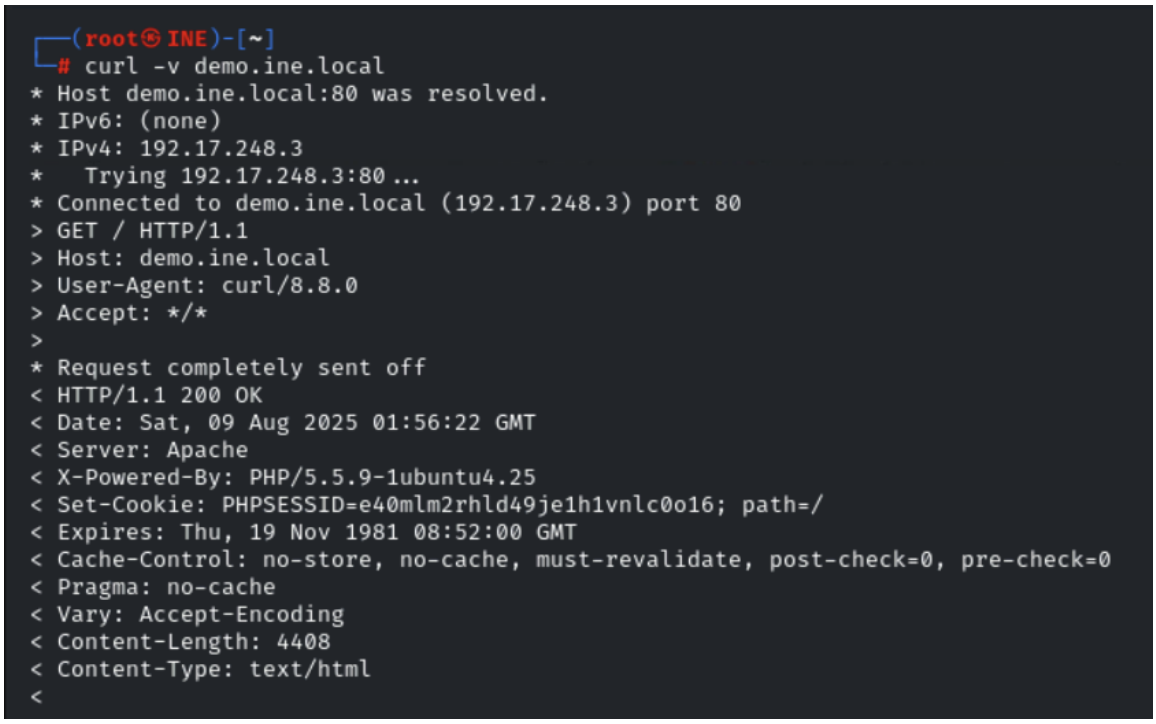
Y finalmente, nuestro sistema o nuestra dirección IP fue enviada de vuelta a la dirección IP del servidor web, junto con el reconocimiento flag ACK.



Entonces después de todas esas conexiones establecidas, podemos ver que se envía la primera solicitud HTTP desde nuestro sistema, o nuestra dirección IP a la dirección IP del servidor web.



Ahora pasemos a una herramienta llamada curl.



En este caso en particular podemos ver que está habilitado el lado del servidor web, nos brinda información muy importante, información que nos puede permitir estructurar nuestros ataques.

También tenemos el lenguaje de programación del backend y el sistema operativo en el que se ejecuta.

Con curl, también tenemos la capacidad de especificar qué tipos de método de solicitud queremos utilizar. Y aquí es donde el nivel más fundamental de modificación de solicitudes HTTP entran en juego.

Entonces, por ejemplo, si quisiéramos aprovechar un método de solicitud HTTP como cabeceras. Y la forma en que podemos hacer esto con curl es: curl –I <http:site.com>

```
┌──(root㉿INE)-[~]
└─# curl -I -v demo.ine.local
* Host demo.ine.local:80 was resolved.
* IPv6: (none)
* IPv4: 192.17.248.3
*   Trying 192.17.248.3:80 ...
* Connected to demo.ine.local (192.17.248.3) port 80
> HEAD / HTTP/1.1
> Host: demo.ine.local
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Date: Sat, 09 Aug 2025 02:08:59 GMT
Date: Sat, 09 Aug 2025 02:08:59 GMT
< Server: Apache
Server: Apache
< X-Powered-By: PHP/5.5.9-1ubuntu4.25
X-Powered-By: PHP/5.5.9-1ubuntu4.25
< Set-Cookie: PHPSESSID=rfraar5tp73klesk85uqq6i034; path=/
Set-Cookie: PHPSESSID=rfraar5tp73klesk85uqq6i034; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
Pragma: no-cache
< Content-Type: text/html
Content-Type: text/html
<

* Connection #0 to host demo.ine.local left intact
```

Ahora, podemos especificar cualquier método de request personalizado que queramos utilizando lo siguiente: curl –I –v –X <parámetro> <http://target.ine.local>

El parametro OPTION dentro de –X sirve para las opciones admitidas para esa página en particular, y como podemos ver las requests aceptadas son GET, HEAD y OPTIONS

```
┌──(root㉿INE)-[~]
└─# curl -I -v -X OPTIONS demo.ine.local
* Host demo.ine.local:80 was resolved.
* IPv6: (none)
* IPv4: 192.17.248.3
*   Trying 192.17.248.3:80 ...
* Connected to demo.ine.local (192.17.248.3) port 80
> OPTIONS / HTTP/1.1
> Host: demo.ine.local
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Date: Sat, 09 Aug 2025 02:12:35 GMT
Date: Sat, 09 Aug 2025 02:12:35 GMT
< Server: Apache
Server: Apache
< X-Powered-By: PHP/5.5.9-1ubuntu4.25
X-Powered-By: PHP/5.5.9-1ubuntu4.25
< Set-Cookie: PHPSESSID=9l47eqng3anostf09egg600fk2; path=/
Set-Cookie: PHPSESSID=9l47eqng3anostf09egg600fk2; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
Pragma: no-cache
< Allow: GET,HEAD,OPTIONS
Allow: GET,HEAD,OPTIONS
< Content-Length: 0
Content-Length: 0
< Content-Type: text/html
Content-Type: text/html
<
```

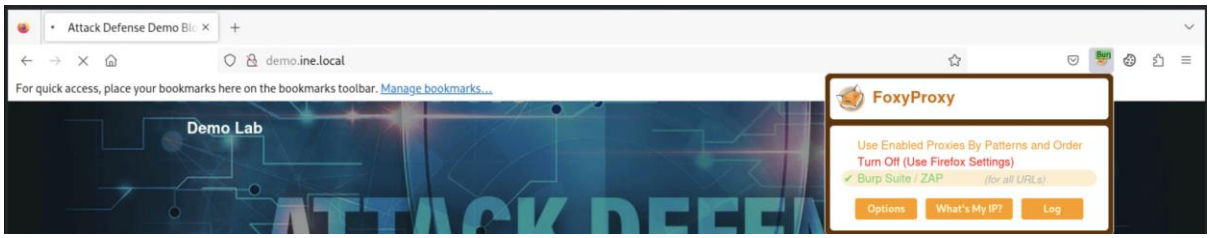
Ahora, por ejemplo, podemos intentar probar un método potencial que es PUT. Este código de estado nos dice “Método no permitido”

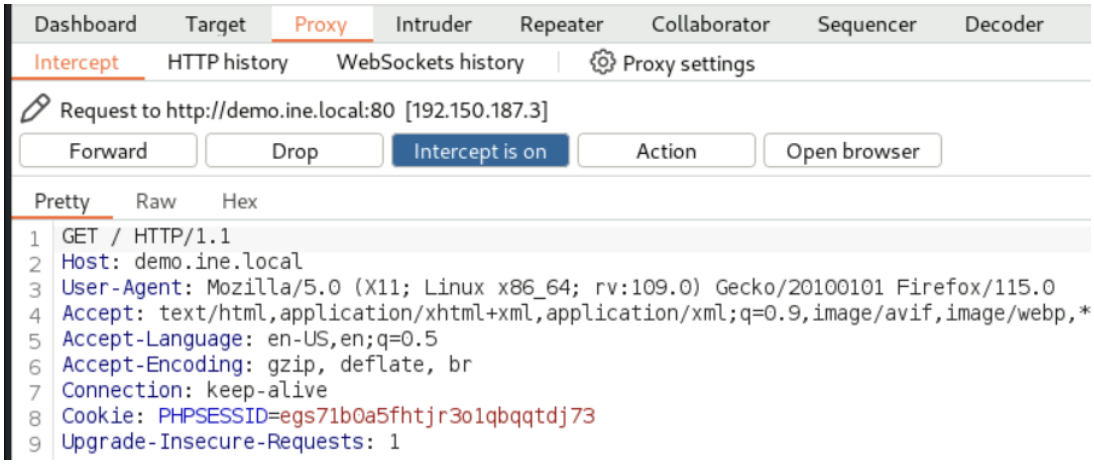
```
┌──(root㉿INE)-[~]
└─# curl -I -v -X PUT demo.ine.local
* Host demo.ine.local:80 was resolved.
* IPv6: (none)
* IPv4: 192.17.248.3
*   Trying 192.17.248.3:80 ...
* Connected to demo.ine.local (192.17.248.3) port 80
> PUT / HTTP/1.1
> Host: demo.ine.local
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 405 Method Not Allowed
HTTP/1.1 405 Method Not Allowed
< Date: Sat, 09 Aug 2025 02:17:13 GMT
Date: Sat, 09 Aug 2025 02:17:13 GMT
< Server: Apache
Server: Apache
< X-Powered-By: PHP/5.5.9-1ubuntu4.25
X-Powered-By: PHP/5.5.9-1ubuntu4.25
< Set-Cookie: PHPSESSID=h9o2he00ip7rs0rnv9d4fthgp1; path=/
Set-Cookie: PHPSESSID=h9o2he00ip7rs0rnv9d4fthgp1; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
Pragma: no-cache
< Content-Length: 221
Content-Length: 221
< Content-Type: text/html
Content-Type: text/html
<

* Closing connection
```

Bien, una vez visto esto, vamos a pasar a la herramienta Burpsuite. Aquí podremos modificar las solicitudes como hemos hecho arriba.

Lo primero que haremos será interceptar la petición de la página web.

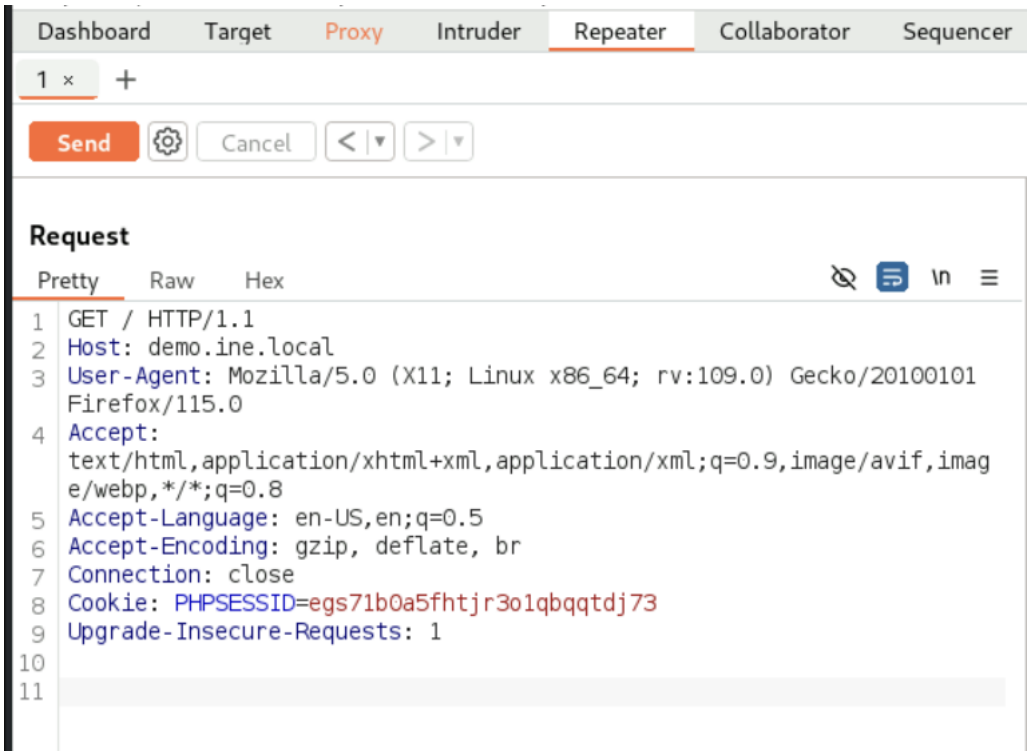




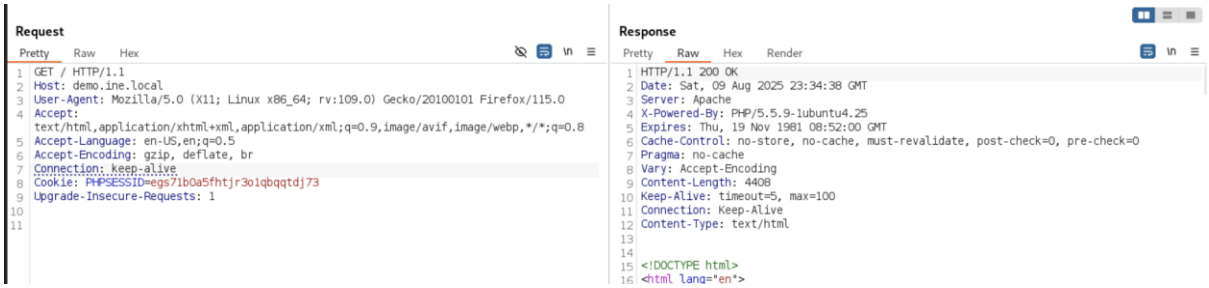
Ahora, lo que quiero enseñarles es un módulo dentro de Burpsuite llamado **Repeater**. El repeater se utiliza para pruebas rápidas o modificaciones de solicitudes para así

poder ver las respuestas muy rápidamente sin la necesidad del navegador web, prácticamente indica cómo se verá una página o qué datos contiene dentro de una respuesta.

Por ejemplo, vamos a cambiar Connection de Close a Keep-alive:



Lo que hará burpsuite es enviar esta solicitud modificada al servidor web y mostrará la respuesta en formato sin formato o en un formato que luego pueda renderizar inmediatamente para ver y realizar pruebas rápidas.



Vamos a probar ahora como se verían las solicitudes a intentos de acceso a recursos compartidos inexistentes

Request

Pretty Raw Hex

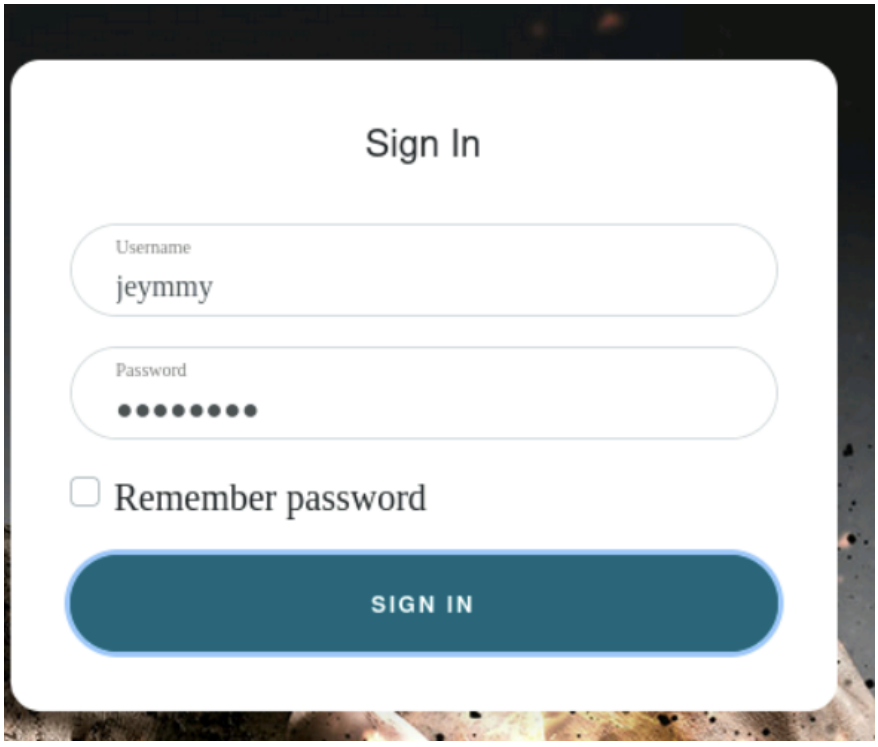
```
1 GET /jeymmy.php HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
  Gecko/20100101 Firefox/115.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif
  ,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: PHPSESSID=egs71b0a5fhtjr3o1qbqqtdj73
9 Upgrade-Insecure-Requests: 1
10
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 404 Not Found
2 Date: Sat, 09 Aug 2025 23:39:21 GMT
3 Server: Apache
4 Content-Length: 208
5 Keep-Alive: timeout=5, max=100
6 Connection: Keep-Alive
7 Content-Type: text/html; charset=iso-8859-1
8
9 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
10 <html>
     <head>
11     <title>
         404 Not Found
```

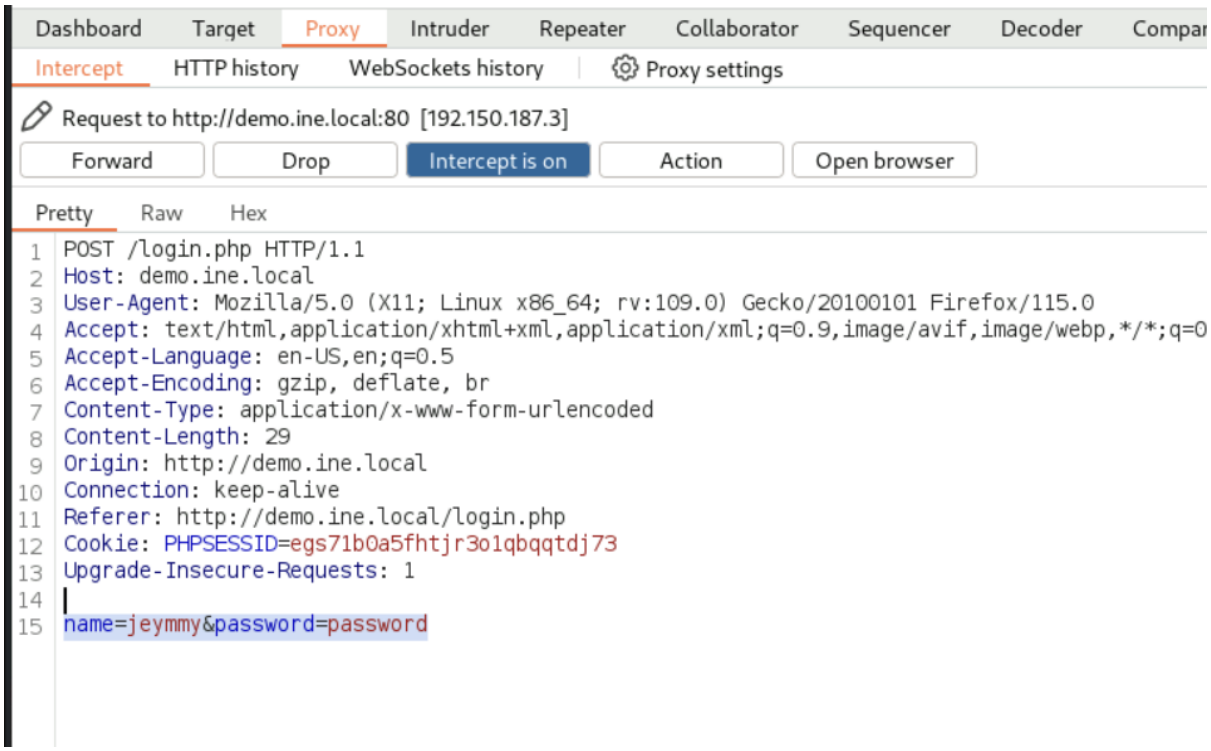
Recordemos que en la página había un apartado de login. Quiero mostrarles cómo exactamente se envían las credenciales a través de HTTP con respecto a qué métodos de solicitud están utilizando y cómo y dónde se colocan las credenciales dentro de la solicitud HTTP.



Entonces, el punto es que puedes ver ahora en lugar de tener un método de solicitud GET, ahora tenemos un método POST, y la URI o el recurso que estamos utilizando que es /login.php.

En términos de que más ha cambiado, se incluye otro encabezado llamado Content-Type que nos dice que tipo de datos se envían. Y lo que es más importante también tenemos Referer que nos dice el servidor web donde se originó esta solicitud en particular antes de que intentásemos logearnos, y por eso se creó desde login.php.

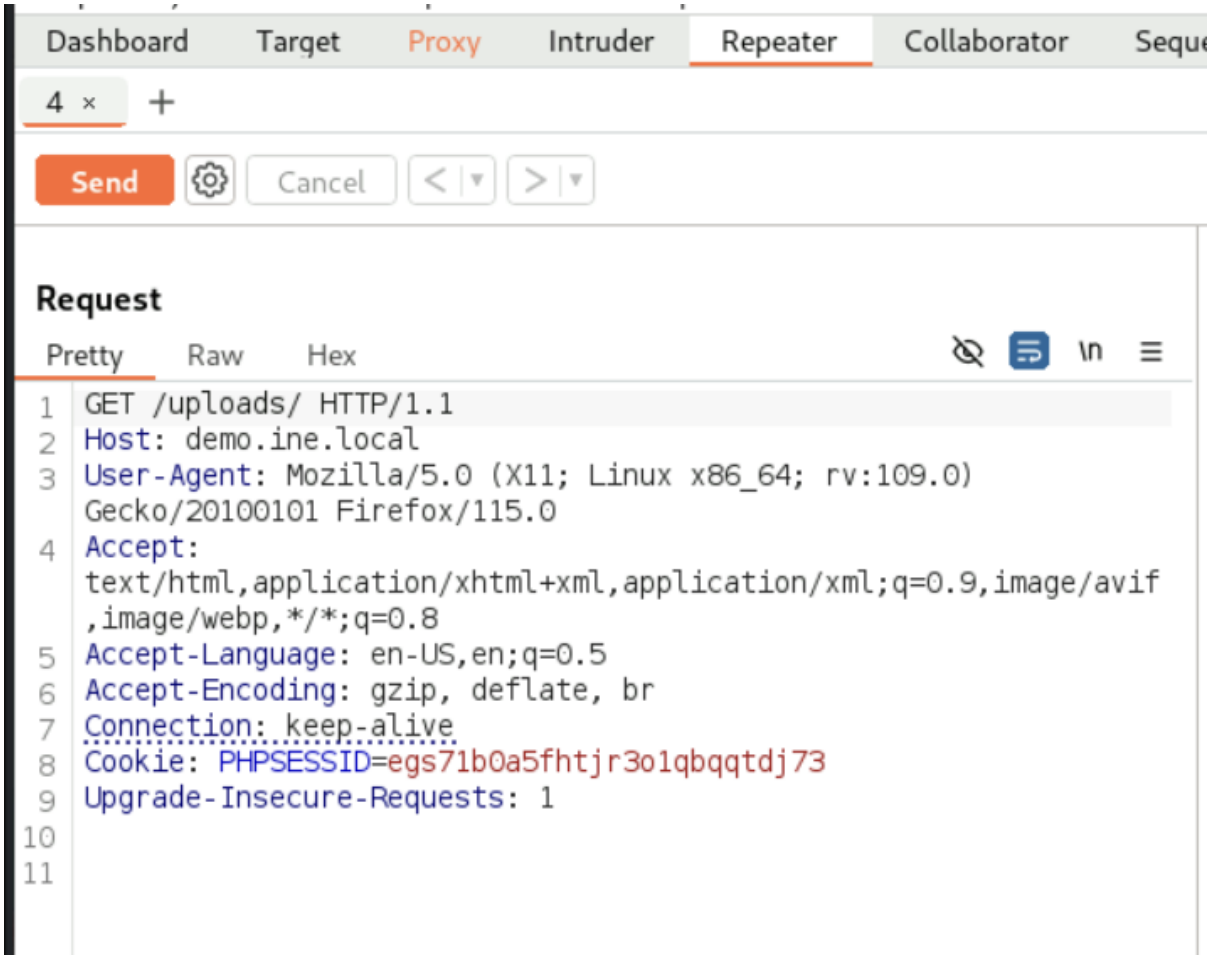
Por último, tenemos las credenciales. Las cuales podemos modificar en el repeater y meterle un diccionario para hacer fuerza bruta.



Una de las cosas que podemos hacer ahora es intentar realizar una fuerza bruta de directorios.

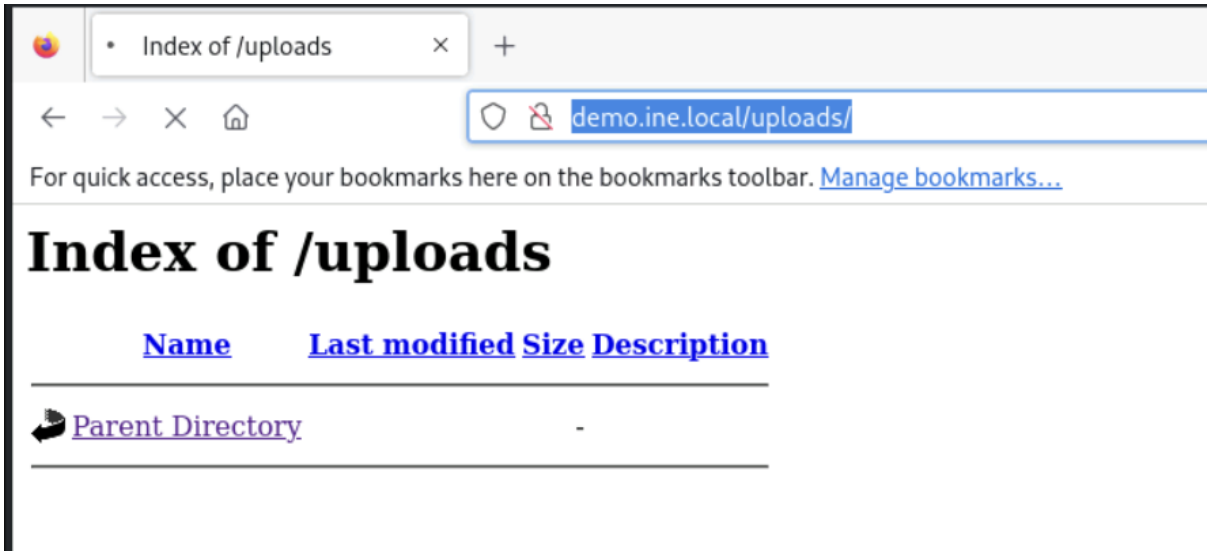
Utilizaremos el directorio Uploads para poner en juego el módulo Repeater de Burpsuite.

```
ry: http://192.150.187.3/uploads/ ——
y IS LISTABLE. No need to scan it.
 you want to scan it anyway)
```

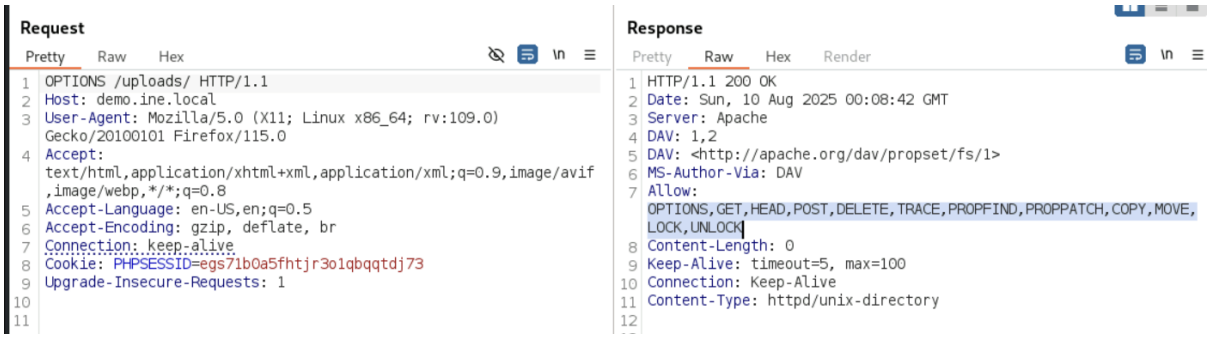


Ahora, lo interesante es, ¿es este directorio en particular admite métodos de solicitud adicionales u opciones HTTP?

Por ejemplo, no podemos acceder a ningún archivo desde ese directorio, pero ¿podríamos subir un archivo o un script maliciosos en este directorio?



Para empezar, modificaremos el método HTTP a OPTIONS para ver las opciones permitidas en el Response.



No nos da la opción para subir archivos, pero prestemos atención a otro encabezado de respuesta. Después del encabezado de Servidor Apache, nos dice que se admite webdav.

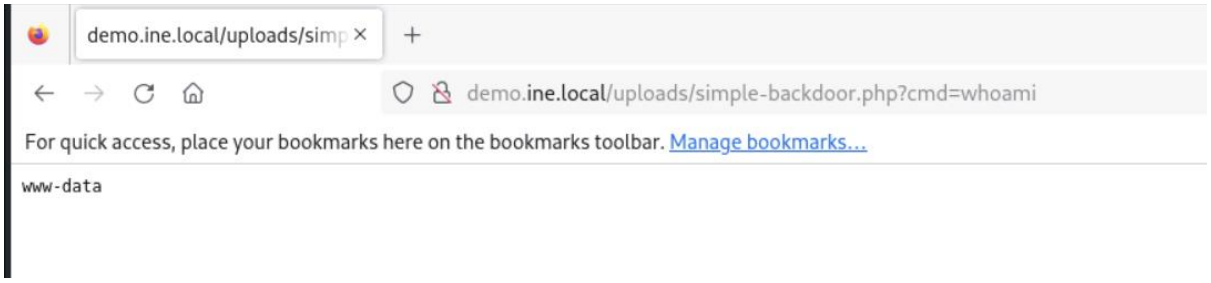
Webdav es un protocolo que permite a los usuarios en la web, compartir, copiar, mover y editar archivos a través de un servidor web, normalmente para documentos.

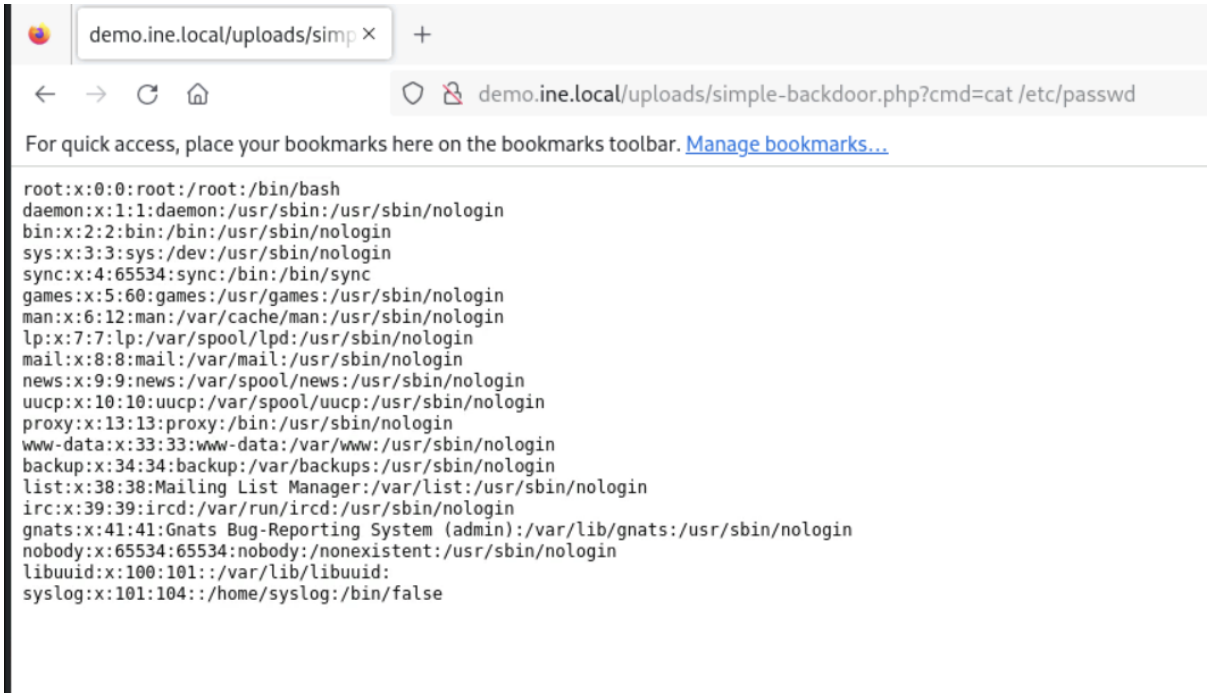
Y la forma en que podemos hacer esto, es descubrir un método como PUT para intentar subir un archivo allí. Aquí es donde entra en juego curl, porque será muy difícil hacerlo manualmente.

Vamos a intentar subir una webshell php, porque recordemos que PHP es el lenguaje del lado del sevidor back-end que se utiliza, lo que significa que cualquier script malicioso o un shell script web debe ser el mismo idioma que usa el servidor web.

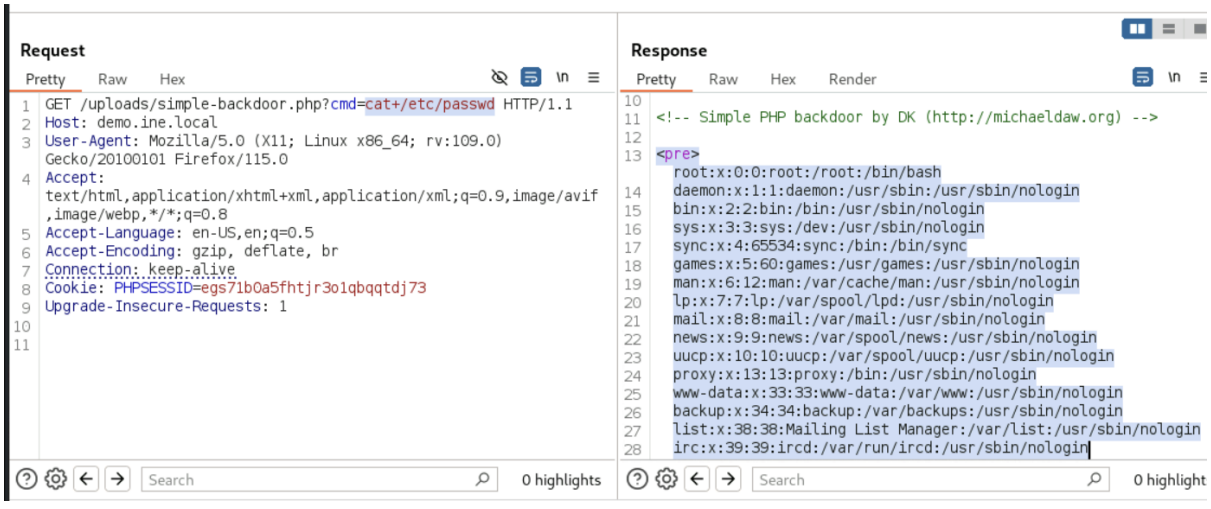
curl http://demo.ine.local/uploads/ --upload-file /usr/share/webshells/php/php-reverse-shell.php

```
┌──(root㉿INE)-[~]
└─# curl http://demo.ine.local/uploads/ --upload-file /usr/share/webshells/php/simple-backdoor.php
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>
┌──(root㉿INE)-[~]
```





Esto también lo podemos hacer desde el Repeater de Burpsuite:



Recordemos que teníamos otros métodos como DELETE, ¿cómo eliminaríamos esa shell que hemos subido?

Request

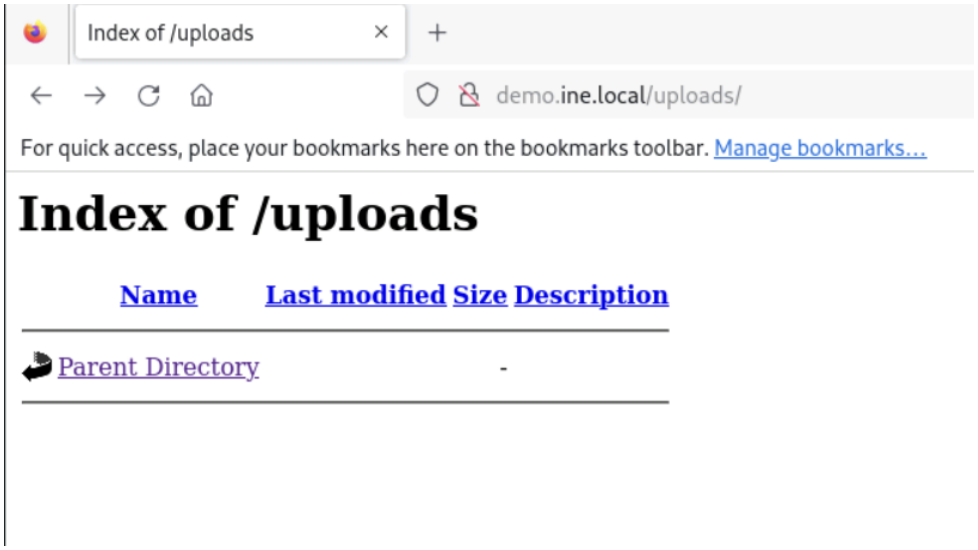
Pretty Raw Hex

```
1 DELETE /uploads/simple-backdoor.php HTTP/1.1
2 Host: demo.ine.local
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
  e/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: PHPSESSID=egs71b0a5fhtjr3o1qbqqtdj73
9 Upgrade-Insecure-Requests: 1
10
11
```

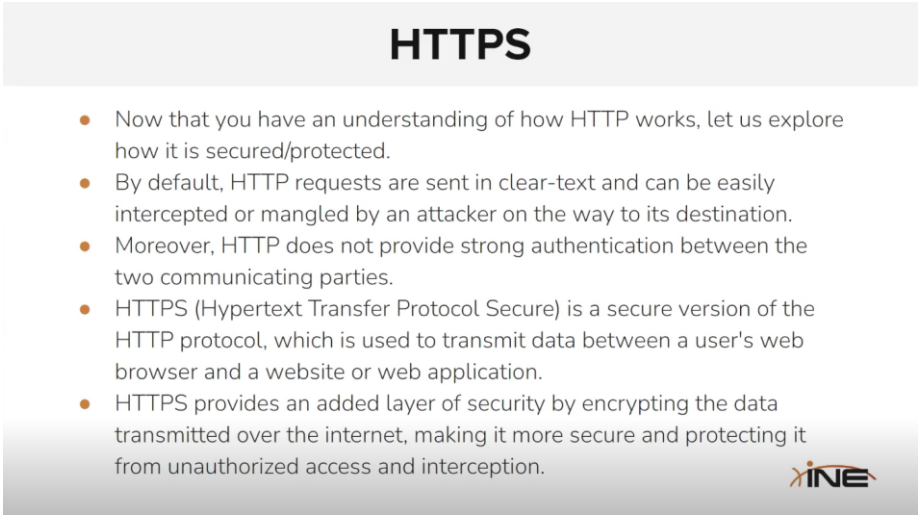
Response

Pretty Raw Hex Render

```
1 HTTP/1.1 204 No Content
2 Date: Sun, 10 Aug 2025 00:36:14 GMT
3 Server: Apache
4 Keep-Alive: timeout=5, max=100
5 Connection: Keep-Alive
6
7
```



HTTPS

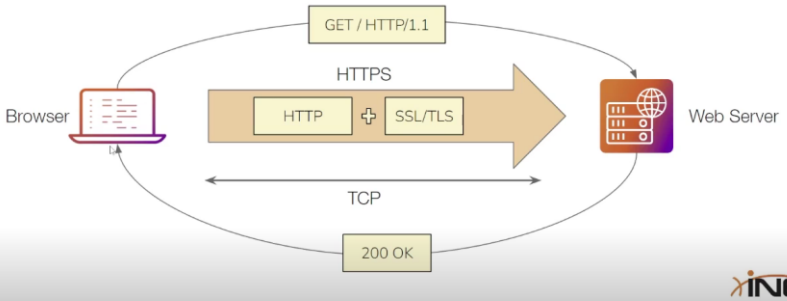


HTTPS

- HTTPS is also commonly referred to as HTTP Secure.
- HTTPS is the preferred way to use and configure HTTP and involves running HTTP over SSL/TLS.
- SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols used to provide secure communication over a computer network, most commonly the internet.
- They are essential for establishing a secure and encrypted connection between a user's web browser or application and a web server.

HTTPS

- This layering technique provides confidentiality, integrity protection and authentication to the HTTP protocol.



HTTPS Advantages

- Encryption of Data in Transit - One of the primary benefits of HTTPS is data encryption during transmission. When data is sent over an HTTPS connection, it is encrypted using strong cryptographic algorithms. This ensures that even if an attacker intercepts the data while it's in transit, they cannot decipher or read its contents.
- Protection Against Eavesdropping - HTTPS prevents unauthorized parties from eavesdropping on the data exchanged between the user's browser and the web server. This is particularly crucial when users input sensitive information, such as login credentials, credit card numbers, or personal details.

IMPORTANTE!!!

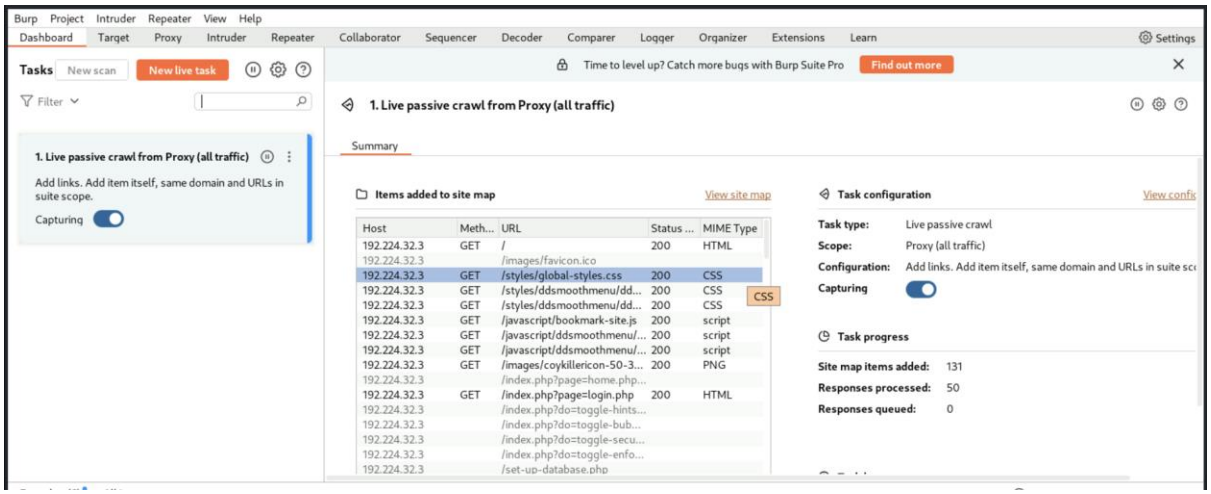
HTTPS Security Considerations

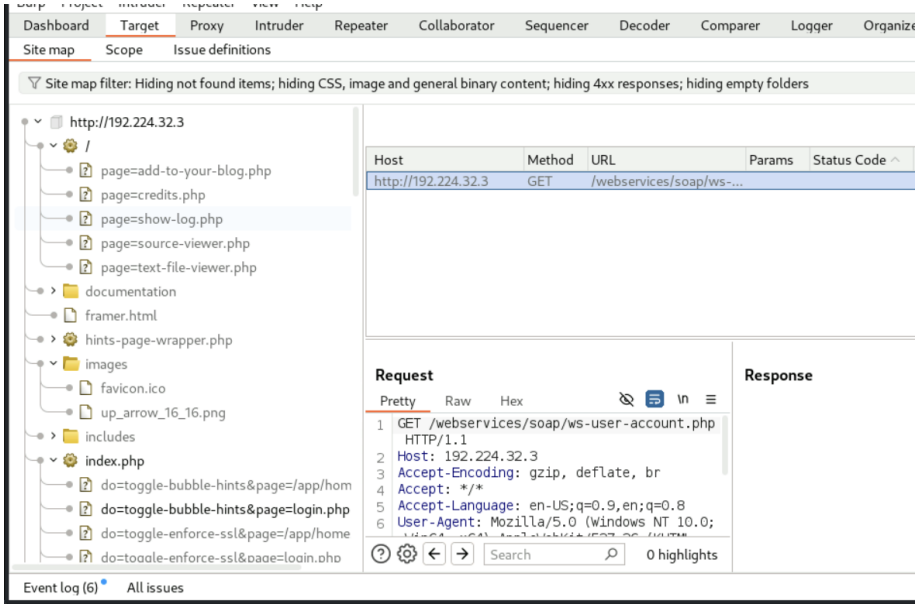
- HTTPS does not protect against web application flaws! Various web application attacks will still work regardless of the use of SSL/TLS. (Attacks like XSS and SQLi will still work)
- The added encryption layer only protects data exchanged between the client and the server and does stop attacks against the web application itself.

Passive Crawling & Spidering with Burp Suite & OWASP ZAP

Vamos a empezar con el passive crawling mediante Burpsuite, pero ¿en qué consiste?

Simplemente haremos una navegación sobre la página web vulnerable y Burpsuite creará un sitemap automáticamente para después analizarlo nosotros.



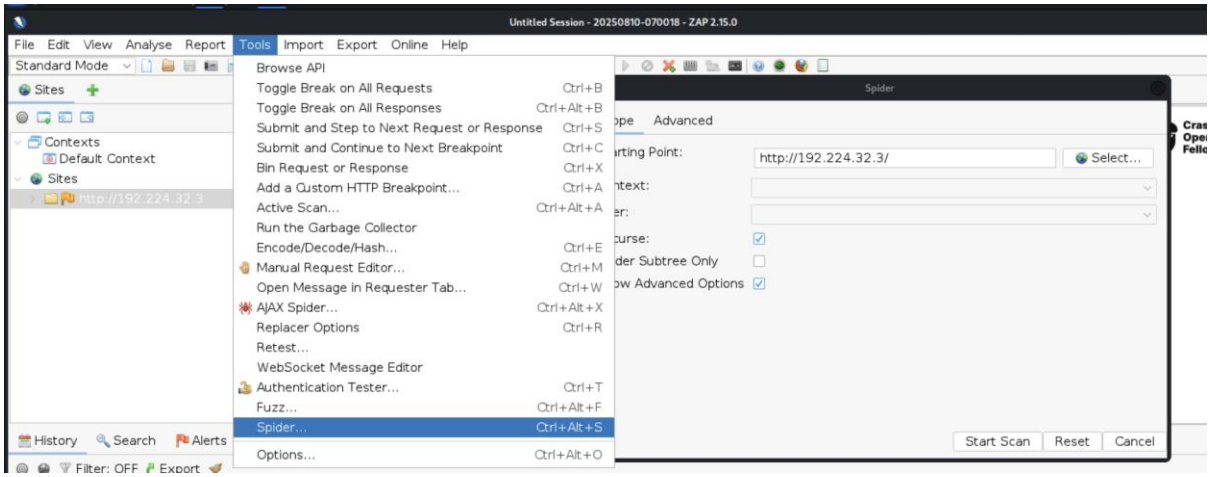


Entonces, basándonos únicamente en lo que hicimos clic, pudimos crear este sitemap y obtener una mejor comprensión de cómo está diseñado y cómo funciona, pero si hacemos esto manualmente sería muy difícil seguirlo.

Ahora echemos un vistazo a OWASP ZAP

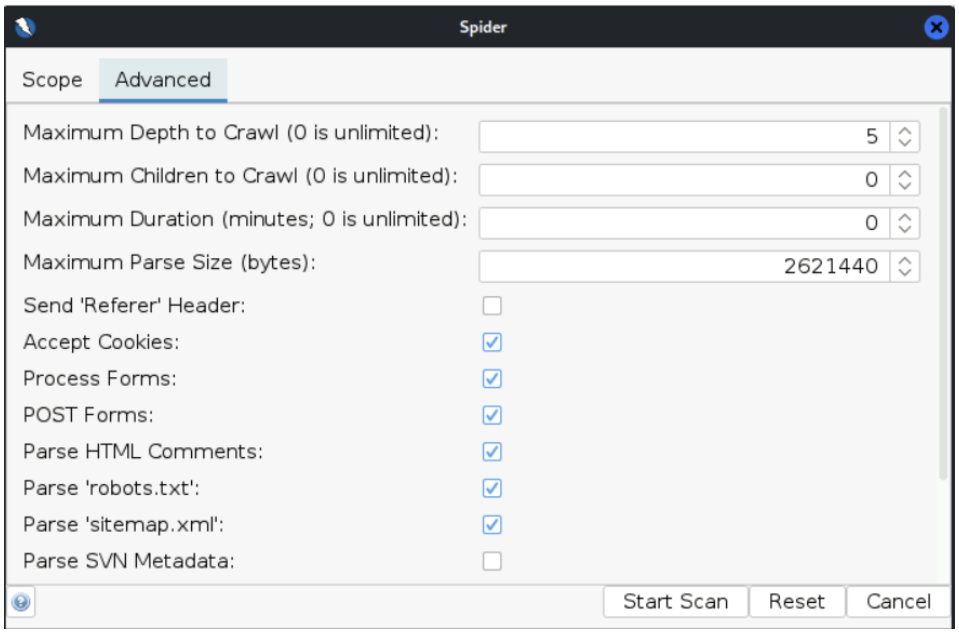
Dejaremos usando el foxy proxy ya que utiliza el puerto 8080 al igual que Burpsuite, eso sí, cerraremos Burpsuite obviamente.

Bueno, una vez dentro de OWASP ZAP vamos a recargar la página.



Una vez interceptado la página, vamos a Tools y seleccionaremos la carpeta que nos ha dado al recargar la página web.

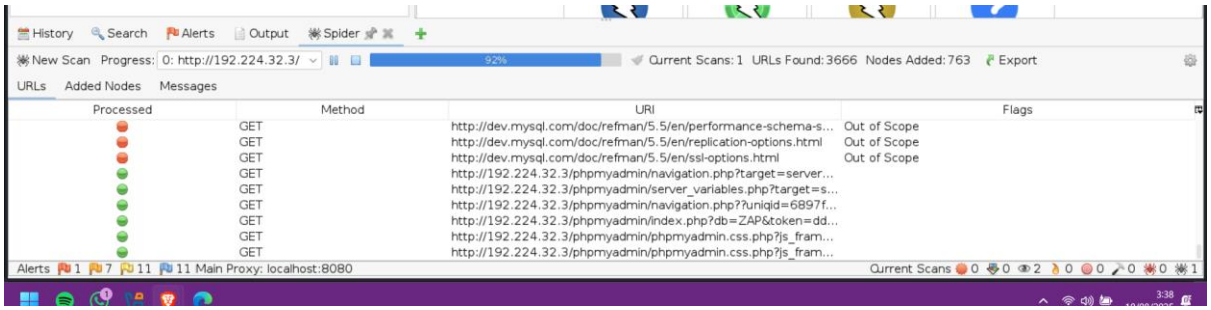
Activaremos la opción “Show Advanced Options” y modificaremos si es necesario, de normal se deja por defecto.

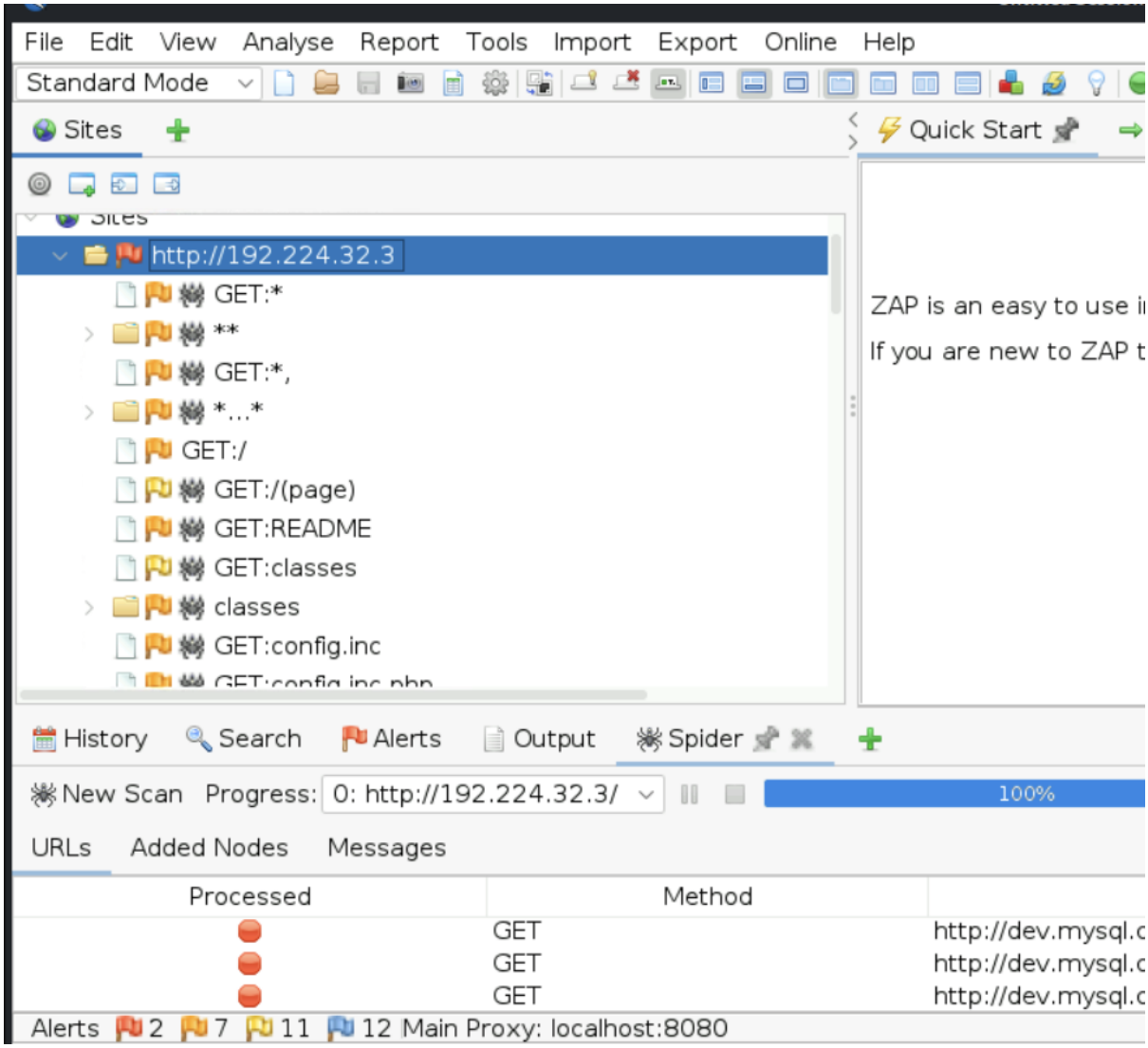


Empezaremos el escaneo.

NOTA: recordemos que esto es un escaneo ACTIVO

Los puntos rojos son hipervínculos externos a sitios como Youtube.com. Los que están en verde son hipervínculos a páginas o directorios para este sitio especifico o para recursos en este servidor en particular.

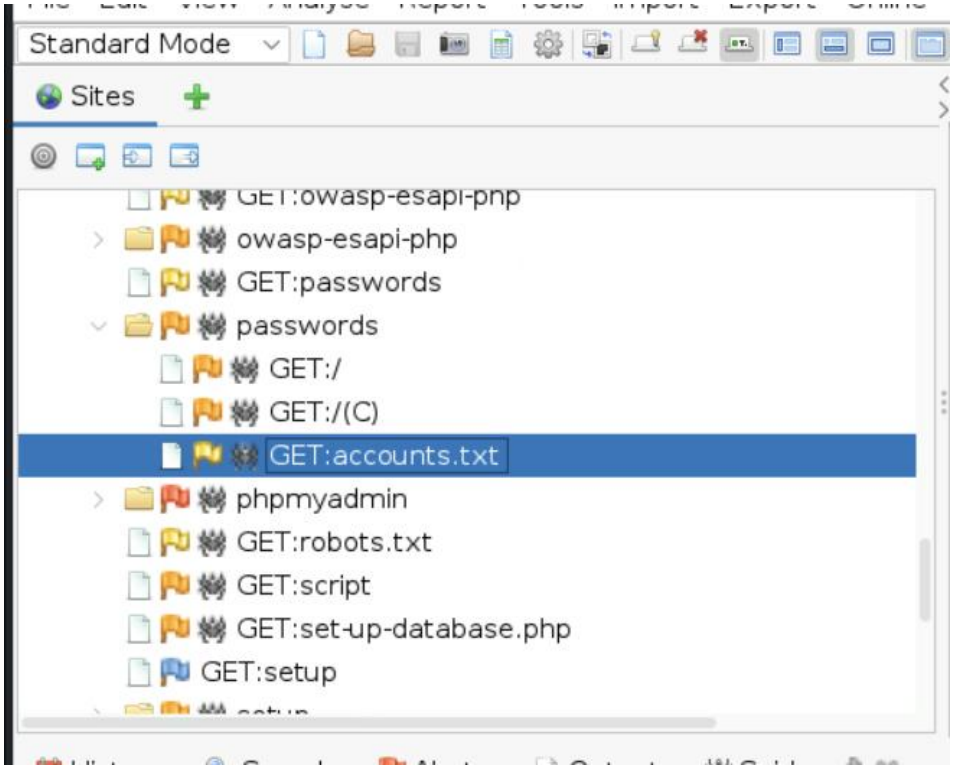




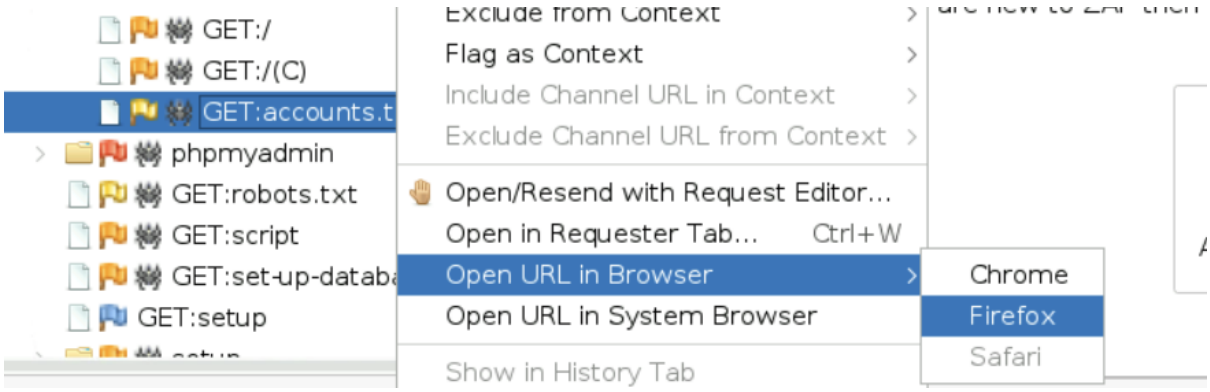
Bien, una vez finalizado el escaneo. Como podemos ver en el mapa del sitio, hemos descubierto una gran cantidad de carpetas y archivos que antes no podíamos identificar ni acceder con Burpsuite, cuando estabamos haciendo Passive Crawling.

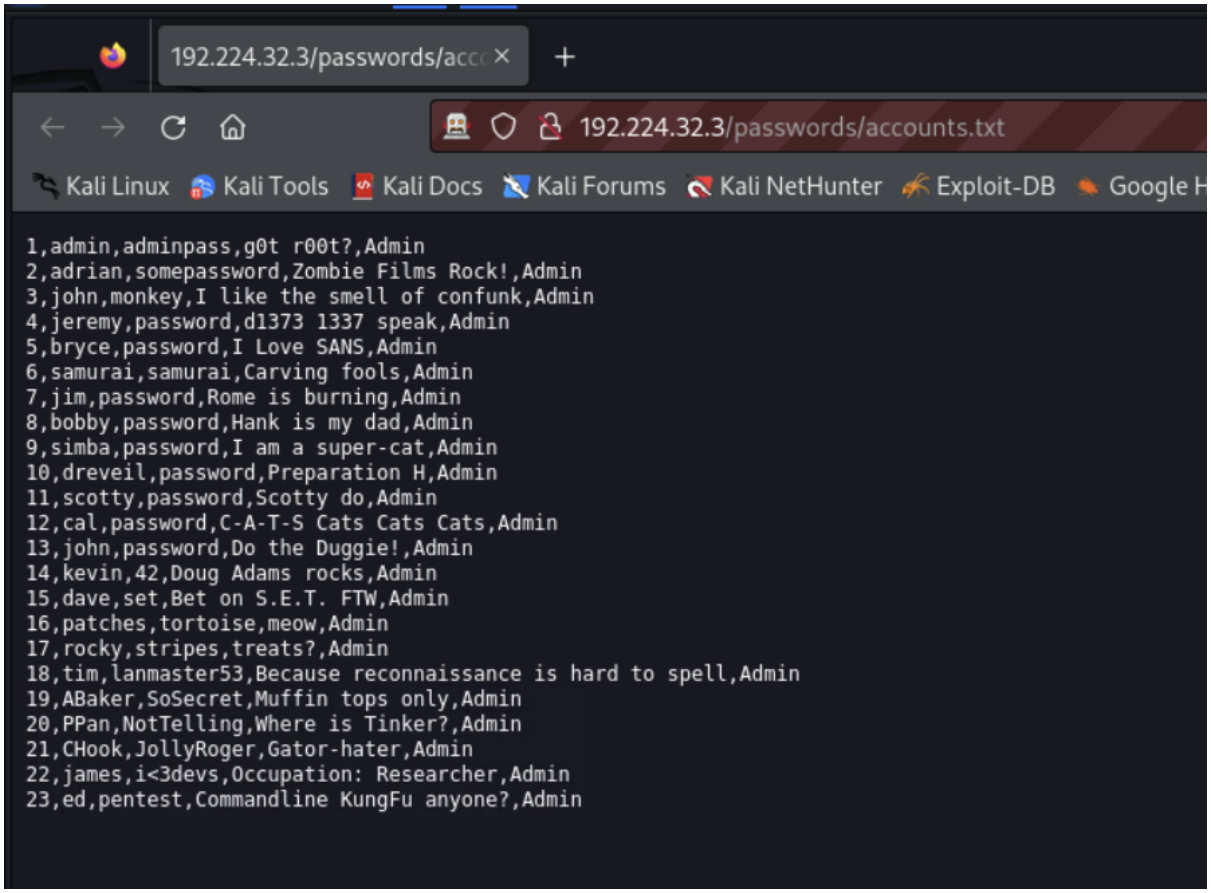
Analicemos un poco lo que hemos encontrado.

Parece que tenemos un archivo llamado accounts.txt. Este es el tipo de información que normalmente no podríamos encontrar porque en el sitio web, si estuviéramos haciendo passive crawling no habría ningún enlace al archivo que contiene accounts.txt



Vamos a abrirlo a ver que contiene.




192.224.32.3/passwords/acc
192.224.32.3/passwords/accounts.txt
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google H
1,admin,adminpass,g0t r00t?,Admin
2,adrian,somepassword,Zombie Films Rock!,Admin
3,john,monkey,I like the smell of confunk,Admin
4,jeremy,password,d1373 1337 speak,Admin
5,bryce,password,I Love SANS,Admin
6,samurai,samurai,Carving fools,Admin
7,jim,password,Rome is burning,Admin
8,bobby,password,Hank is my dad,Admin
9,simba,password,I am a super-cat,Admin
10,dreveil,password,Preparation H,Admin
11,scotty,password,Scotty do,Admin
12,cal,password,C-A-T-S Cats Cats Cats,Admin
13,john,password,Do the Duggie!,Admin
14,kevin,42,Doug Adams rocks,Admin
15,dave,set,Bet on S.E.T. FTW,Admin
16,patches,tortoise,meow,Admin
17,rocky,stripes,treats?,Admin
18,tim,lanmaster53,Because reconnaissance is hard to spell,Admin
19,ABaker,SoSecret,Muffin tops only,Admin
20,PPan,NotTelling,Where is Tinker?,Admin
21,CHook,JollyRoger,Gator-hater,Admin
22,james,i<3devs,Occupation: Researcher,Admin
23,ed,pentest,Commandline KungFu anyone?,Admin