Filter Evasion & WAF Bypass Techniques

Course Overview

Alexis Ahmed

Offensive Security/Red Team Instructor @INE
Red Team Lead @HackerSploit

Key Concepts

+ Data Encoding Basics
+ Server-Side & Client-Side Filter Evasion
+ WAF & Proxy Bypass Techniques

MAJOR TOPICS

- Data Encoding Fundamentals (HTML, URL, Base64 etc)
- Input Filtering Fundamentals
- Client-Side & Server-Side Filter Evasion (XSS, SQLi etc)
- WAF & Proxy Bypass Techniques

LEARNING OUTCOMES

+ 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 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.

PREREQUISITES

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

LET'S GO!

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	&#32;		Space
33	041	21	00100001	!	&#33;	&excl;	Exclamation mark
34	042	22	00100010	"	&#34;	&quot;	Double quotes (or speech marks)
35	043	23	00100011	#	&#35;	&num;	Number sign
36	044	24	00100100	$	&#36;	&dollar;	Dollar
37	045	25	00100101	%	&#37;	&percnt;	Per cent sign
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	'	&#39;	&apos;	Single quote
40	050	28	00101000	(	&#40;	&lparen;	Open parenthesis (or open bracket)
41	051	29	00101001	)	&#41;	&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)
-   for a non-breaking space
- — 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= hello world
Firefox	arg=test	arg=%20test%20with%20spaces	arg=%3Ch1%3Ehello%20world%3C/h1%3E
IE	arg=test	arg= test with spaces	arg= hello world
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

Filter Evasion & WAF Bypass Techniques

Course Summary

Key Concepts - Recap

+ Data Encoding Basics
+ Server-Side & Client-Side Filter Evasion
+ WAF & Proxy Bypass Techniques

Learning Outcomes - Recap

+ 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 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.

Real-World Applications

- Modern applications deploy security measures like input validation, blacklists/whitelists, and WAFs to filter malicious inputs. The ability to bypass these mechanisms allows testers to accurately simulate sophisticated attack scenarios used by real-world adversaries.
- The ability to successfully evade filters or bypass WAF rules, allows you to evaluate the robustness of an application's security measures and provide actionable recommendations for improvement.
- Vulnerabilities like SQL Injection, XSS, and SSRF may not be exploitable with simple payloads due to filters. Learning evasion techniques helps testers circumvent these protections to exploit the vulnerabilities fully.

Next Steps

- Research and practice bypassing popular WAFs such as:
  - ModSecurity
  - Cloudflare
  - Imperva
- Explore WAF bypass techniques like:
  - Payload manipulation
  - Traffic shaping (e.g., sending payloads in chunks)
  - Using alternative encoding or non-standard request

THANKS FOR WATCHING!

EXPERTS AT MAKING YOU AN EXPERT