

Module 1 - Encoding and Filtering

Data Encoding Basics

Encoding Types

- URL encoding
- HTML encoding
- Base (32/64) encoding
- Unicode encoding

URL encoding

- contains characters in the range of the US-ASCII code char set.
- it's also called percent-encoding
- it replaces characters outside the allowed set with a % followed by the two hexadecimal digits representing the numeric value of the octet.

CLASSIFICATION	INCLUDED CHARACTERS	ENCODING REQUIRED?
Safe characters	Alphanumeric [0-9a-zA-Z], special characters \$-_+!*(), and reserved characters used for their reserved purposes (e.g., question mark used to denote a query string)	NO
ASCII Control characters	Includes the ISO-8859-1 (ISO-Latin) character ranges 00-1F hex (0-31 decimal) and 7F (127 decimal.)	YES
Non-ASCII characters	Includes the entire "top half" of the ISO-Latin set 80-FF hex (128-255 decimal.)	YES
Reserved characters	\$ & + , / ; = ? @ (not including blank space)	YES*
Unsafe characters	Includes the blank/empty space and " < > # % { } \ ^ ~ [] "	YES

Some commonly encoded chars

CHARACTER	PURPOSE IN URI	ENCODING
#	Separate anchors	%23
?	Separate query string	%3F
&	Separate query elements	%24
%	Indicates an encoded character	%25
/	Separate domain and directories	%2F
+	Indicates a space	%2B
<space>	Not recommended	%20 or +

HTML encoding

Document character encoding

Define char encoding using http :-

HTTP header → Content-Type:

- if this header is sent, we will see something like → Content-Type: text/html; charset=utf-8
- If this header is not defined, the RFC defines as the default charset the ISO-8859-1

Define character encoding using HTTP

PHP> Uses the header() function to send a raw **HTTP** header:
header('Content-type: text/html; charset=utf-8');

ASP.Net> Uses the response object:

<%Response.charset="utf-8"%>

JSP> Uses the page directive:

<%@ page contentType="text/html; charset=UTF-8" %>

Define character encoding using HTTP

It is also possible to set the character encoding using the **HTML** directive **META**. For example, this code is useful in specifying the character encoding of the current document to **UTF-8**:

```
<meta http-equiv="Content-Type" Content="text/html; charset=utf-8">
```

With **HTML5**, is also possible to write: `<meta charset="utf-8">`

Character Reference	Rule	Encoded character
Named entity	& + <u>named character references</u> + ;	<
Numeric Decimal	& + # + D + ; D = a decimal number	<
Numeric Hexadecimal	& + #x + H + ; H = an hexadecimal number (case-insensitive)	< <

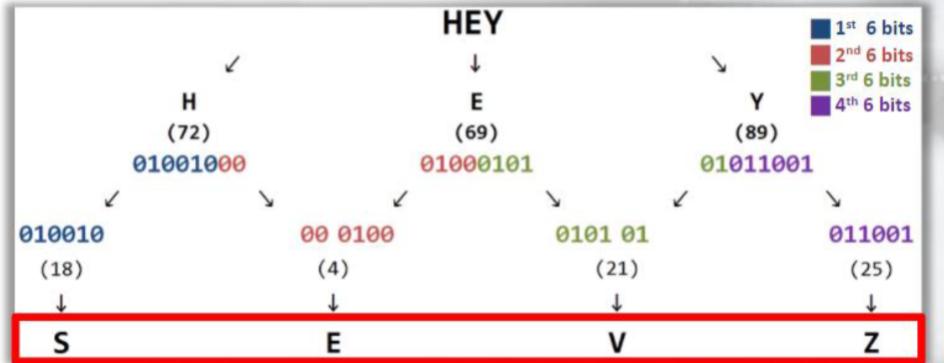
- HTML entities → <https://www.freeformatter.com/html-entities.html>
- ###Hexadecimal aka base16###

Base(36-64)

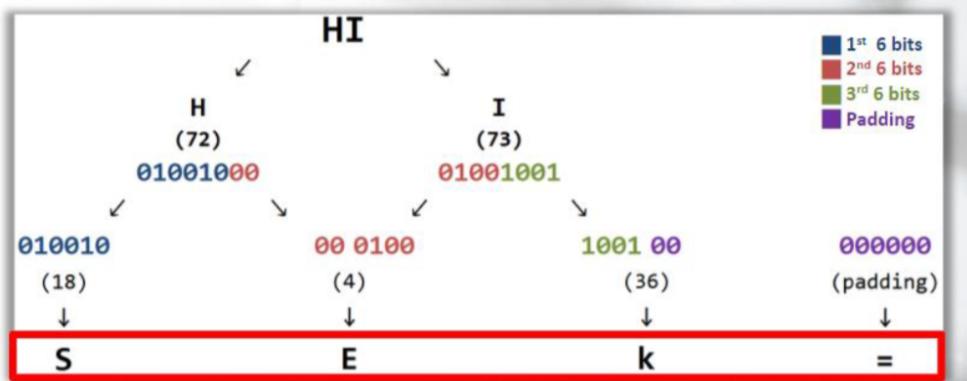
- base10 is case insensitive
- base36 is used in many real world scenarios

Base 64 Encoding Scheme

To encode the term "HEY" we have:



To encode the term "HI" we have:



Base 64 Encoding Scheme: JavaScript

Many browsers can handle **Base64** natively through functions **btoa** and **atob**:

```
window.btoa('encode this string'); //Encode
```

```
window.atob('ZW5jb2RlIHRoaXMgc3RyaW5n'); //Decode
```

Unicode encoding

homoglyph/Visual spoofing attack generator

- <https://www.irongeek.com/homoglyph-attack-generator.php>

There 3 ways to map Unicode character points:

- UTF-8
- UTF-16
- UTF-32

UTF → Unicode Transformation Format

Punycode and Homoglyph attacks to obfuscate URLs for Phishing

<https://www.irongeek.com/i.php?page=security/out-of-character-use-of-punycode-and-homoglyph-attacks-to-obfuscate-urls-for-phishing>

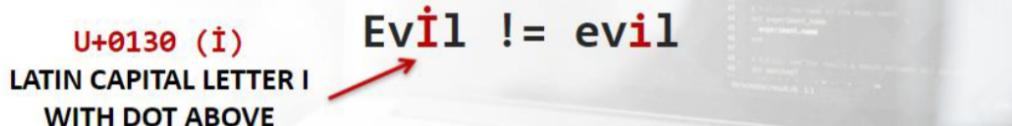
Computer Interpretations - Example: Censored Feedback

The input flow could be as follow:

⌚ A user sends the following message:

Evİl intent, as usual!

⌚ The filter checks for evil strings, but without success.



Unicode Security Guide

- <https://websec.github.io/unicode-security-guide/>

Computer Interpretations: Mixed Examples

Normalization:

①②③④ ⑤⑥⑦⑧ **becomes** drop table

Canonicalization:

⠄ (U+2039) ⠄ (U+FE64) ⠄ (U+ff1c) } **becomes** ⠄ (U+003C)

- <https://codepoints.net/>
- <https://qaz.wtf/u/convert.cgi>

Multiple (De|En) codings

Filtering Basics

Best Practice to Protect Web Apps

- Input Filtering and Output Encoding controls are crucial.

Controls can be implemented at different layers in a web application. They can be represented as either **libraries and APIs** (by naive developers) or, in the best case, by internal specialists or external organizations, like **ESAPI by OWASP**.

Security controls are also inside most common browsers.

ESAPI by OWASP

- A free, open-source web application security control library that helps programmers write lower-risk applications.

- **Best Practice:** Use external solutions for web app protection, such as WAFs, instead of internal solutions like libraries or APIs.
- **Examples:** Paid and free open-source WAF software, such as **ModSecurity** (free WAF).

WAF

- Instructions on what a WAF or filter must block/allow are defined as **rules** (also referred to as **filters**).

RegEx

- **Two Main Types of Regex Engines:**
 - **DFA (Deterministic Finite Automaton):** Faster due to its deterministic approach but lacks features like **lazy quantifiers** and **backreferences**.
 - **NFA (Nondeterministic Finite Automaton):** More popular and supports additional regex features.
- https://en.wikipedia.org/wiki/Comparison_of_regular_expression_engines

ENGINE	PROGRAM
DFA	awk, egrep, MySQL, Procmail
NFA	.NET languages, Java, Perl, PHP, Python, Ruby, PCRE library, vi, grep, less, more

Metacharacters

Char	Name	Meaning
^	Caret	The position at the START of the line.
\$	Dollar	The position at the END of the line.
()	Opening/Closing parenthesis	Start/close a characters group.
[]	Opening/Closing square bracket	Start/close a characters class.
?	Question mark	One or zero (optional) of the immediately-preceding item (char or group).
+	Plus	One or more of the immediately-preceding item (char or group).
*	Star or asterisk	Any number, including none, of the immediately-preceding item (char or group).
.	Period or dot	Shorthand for a character class that matches any character.
\	Backslash	Escape special characters.
	Vertical bar or pipe	It means OR. Combines multiple expressions in one that matches any of single ones.
{}	Opening/Closing curly brace	Start/close repetitions of a characters class.

Shorthand Character Classes

SHORTHAND	NAME	MEANING
<code>^</code>	Caret	If at the beginning of the character class, it means to reverse the matching for the class.
<code>\d</code>	Digit	Matches any digit character. The same as <code>[0-9]</code>
<code>\D</code>	Non-digit	The complement of \d. The same as <code>[^\d]</code>
<code>\w</code>	Part-of-word character	Matches any alphanumeric character or an underscore. The same as <code>[a-zA-Z0-9_]</code> In some flavors the underscore is omitted.
<code>\W</code>	Non-word character	The complement of \w. The same as <code>[^\w]</code>
<code>\s</code>	Whitespace character	Matches any whitespace character. The same as <code>[\f\n\r\t\v]</code>
<code>\S</code>	Non-whitespace character	The complement of \s. The same as <code>[^\s]</code>

Non-Printing Characters

used for obfuscating the payload

SHORTHAND	NAME (Symbol, Unicode)	MEANING
<code>\0</code>	NUL (█ U+0000)	NUL Byte, in many programming languages marks the end of a string.
<code>\b</code>	Backspace (█ U+0008)	Within a character class represent the backspace character, while outside \b matches a word boundary.
<code>\t</code>	Horizontal tab (█ U+0009)	Generated by the Tab key on a standard keyboard.
<code>\n</code>	Line feed (█ U+000A)	New line.
<code>\v</code>	Vertical tab (█ U+000B)	Vertical tabulation.
<code>\f</code>	Form feed (█ U+000C)	Form feed.
<code>\r</code>	Carriage return (█ U+000D)	In HTTP, the \r\n sequence is used as the end-of-line marker.
<code>\e</code>	Escape (█ U+001B)	Escape character (Only for GNU Compiler Collection).

Unicode

Match Unicode Code Point

For example, the regex `\u2603` matches the snowman character  in .NET, Java, JavaScript and Python.

If we want to match the same character to the PCRE library in Apache and PHP, we must use the other notation:

`\x{2603}`

CHARACTER QUALITY	DESCRIPTION
<code>\p{L} or \p{Letter}</code>	All the letters, from any language.
<code>\p{LI} or \p{Lowercase_Letter}</code>	Lowercase letters that have the respective uppercase quality.
<code>\p{Z} or \p{Separator}</code>	Characters used to separate, but without visual representation.
<code>\p{S} or \p{Symbol}</code>	Currency symbols, math symbols, etc...
<code>\p{N} or \p{Number}</code>	All the numeric characters.
<code>\p{Nd} or \p{Decimal_Digit_Number}</code>	Numbers from zero to nine in multiple scripts except Chinese, Japanese, and Korean.
<code>\p{P} or \p{Punctuation}</code>	All the punctuation characters.

WAF - Web Application Firewall

- **Regex** is the main method to define how a WAF should behave.
- **Whitelisting mode** is the best solution to protect a web application.
 - **Customization:** Rules customization is challenging because it requires deep knowledge of both the application and the WAF solution.
 - **Drawback:** Whitelisting is prone to false positives, which is why WAFs are often deployed in **blacklisting mode** instead.

Blacklisting Mode

- Involves a collection of well-known attacks; WAF producers create a list of rules to protect against common vulnerabilities.

- **Drawback:** Multiple ways to achieve the same attack goal mean every small change in the attack payload must be added to the blacklist. Failure to do so can result in a **WAF bypass**.
- **Note: All WAFs can be bypassed!**

Simple Rules to Bypass WAFs

- **XSS:**
 - Instead of: `alert('xss')` or `alert(1)`
 - Use: `prompt('xss')`, `prompt(8)`, `confirm(8)`, `confirm('XSS')`, `alert(/xss/.source)`, `window`
 - Instead of: `alert(document.cookie)`
 - Use: `with(document)alert(cookie)`, `alert(document['cookie'])`, `alert(document[/cookie/.source])`, `alert(document[/coo/.source+/kie/.source])`
 - Instead of: ``
 - Use: `<svg/onload=alert(1)>`, `<video src=x onerror=alert(1);>`, `<audio src=x onerror=alert(1);>`, `data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4=(<script>alert('xss')</script>)`
- **Blind-SQLi:**
 - Instead of: `' or 1=1`
 - Use: `' or 6=6`, `' or 0x47=0x47`, `or char(32)=''`, `or 6 is not null`
- **SQLi:**
 - Instead of: `UNION SELECT`
 - Use: `UNION ALL SELECT`
- **Directory Traversal:**
 - Instead of: `/etc/passwd`
 - Use: `/too/./etc/far/./passwd`, `/etc//passwd`, `/etc/ignore/./passwd`, `/etc/passwd.....`
- **Web Shell:**
 - Instead of: `c99.php`, `r57.php`, `shell.aspx`, `cmd.jsp`, `CmdAsp.asp`

- Use: `augh.php`

WAF Detection and Fingerprinting

- WAFs usually operate in **passive** mode, **reactive** mode, or both.
 - **Passive Mode:** Reduces the number of false positives and avoids blocking the application.
 - **Reactive Mode:** Most WAFs in production are in this mode.

Before Testing a Web App, it's extremely useful to know if there is a WAF and its type.

- **Tools:**

- **wafw00f** is the best tool for WAF detection.
- **nmap**: Use the script `http-waf-fingerprint` for additional detection.

```
ohpe@kali:/$ nmap --script=http-waf-fingerprint www.imperva.com -p 80
Starting Nmap 6.40 ( http://nmap.org ) at 2014-04-08 14:20 CEST
Nmap scan report for www.imperva.com (185.11.125.104)
Host is up (0.045s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-waf-fingerprint:
|   Detected WAF
|_   Incapsula WAF

Nmap done: 1 IP address (1 host up) scanned in 11.57 seconds
```

Another interesting resource is [imperva-detect](#) by Lamar Spells. This utility is 100% focused on the detection of an Imperva WAF and it runs 6 tests, one baseline and five additional:

```
# Test 0 - Baseline to establish expected behavior
# Test 1 - "Web Leech" blocking
# Test 2 - "E-mail Robot" blocking
# Test 3 - BlueCoat Proxy Manipulation blocking
# Test 4 - Web Worm blocking
# Test 5 - XSS blocking
```

XSS Filter

XSS Filter (Internet Explorer):

XSS Filter is enabled by default in the Internet, Trusted, and Restricted security zones, but an interesting feature was introduced **to disable the filter.**

The main reason was because some sites may depend on the reflected values that the filter searches for.

X-XSS-Protection: 0

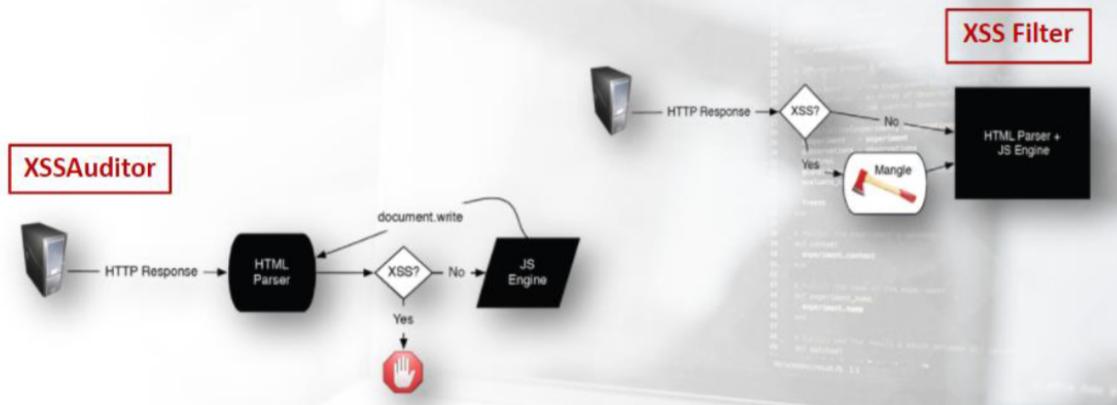
XSS Filter (Internet Explorer):

Later, the IE team added support to a new token in the X-XSS-Protection header:

`X-XSS-Protection: 1; mode=block`

With this token, if a potential reflected XSS attack is detected, the browser, rather than attempting to sanitize the page, will render a simple #. Here is a **simple test.**

XSSAuditor (WebKit/Blink): XSS Filter vs XSSAuditor



What is the best technique to evade XSS filters?

^

You can find more examples in the OWASP resource based on the XSS Cheat Sheet by RSnake.

- Using Character Encoding. ...
- Case Manipulation and Character Insertion. ...
- Fooling Regular Expressions. ...
- Using Atypical Event Handlers. ...
- Tricks with Tags and Attributes. ...
- Using Atypical Delimiters. ...
- Making Other Deliberate Mistakes.