

Encoding/Decoding

As we modify and send custom HTTP requests, we may have to perform various types of encoding and decoding to interact with the webserver properly. Both tools have built-in encoders that can help us in quickly encoding and decoding various types of text.

URL Encoding

It is essential to ensure that our request data is URL-encoded and our request headers are correctly set. Otherwise, we may get a server error in the response. This is why encoding and decoding data becomes essential as we modify and repeat web requests. Some of the key characters we need to encode are:

- **Spaces:** May indicate the end of request data if not encoded
- **&:** Otherwise interpreted as a parameter delimiter
- **#:** Otherwise interpreted as a fragment identifier

To URL-encode text in Burp Repeater, we can select that text and right-click on it, then select **(Convert Selection>URL>URL encode key characters)**, or by selecting the text and clicking **[CTRL+U]**. Burp also supports URL-encoding as we type if we right-click and enable that option, which will encode all of our text as we type it. On the other hand, ZAP should automatically URL-encode all of our request data in the background before sending the request, though we may not see that explicitly.

There are other types of URL-encoding, like **Full URL-Encoding** or **Unicode** URL encoding, which may also be helpful for requests with many special characters.

Decoding

While URL-encoding is key to HTTP requests, it is not the only type of encoding we will encounter. It is very common for web applications to encode their data, so we should be able to quickly decode that data to examine the original text. On the other hand, back-end servers may expect data to be encoded in a particular format or with a specific encoder, so we need to be able to quickly encode our data before we send it.

The following are some of the other types of encoders supported by both tools:

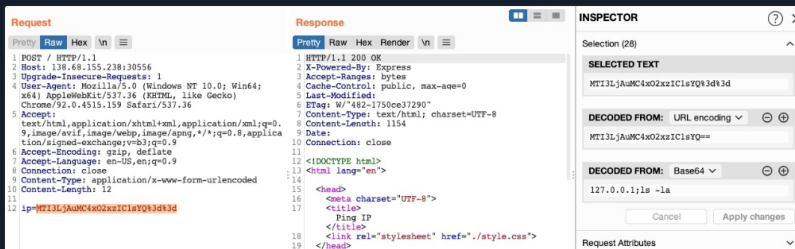
- HTML
- Unicode
- Base64
- ASCII hex

To access the full encoder in Burp, we can go to the **Decoder** tab. In ZAP, we can use the **Encoder/Decoder/Hash** by clicking **[CTRL+E]**. With these encoders, we can input any text and have it quickly encoded or decoded. For example, perhaps we came across the following cookie that is base64 encoded, and we need to decode it: **eyJ1c2VybmFtZSI6ImdlZXN0IiwgImIzX2FkbWUiOiJpmYWxzZX0=**

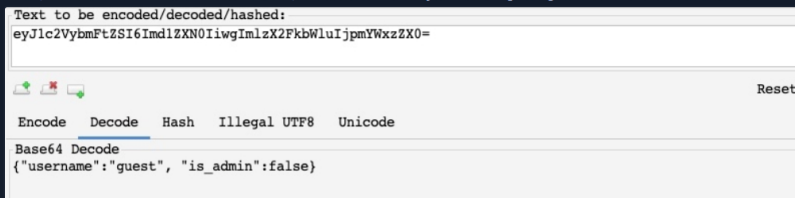
We can input the above string in Burp Decoder and select **Decode as > Base64**, and we'll get the decoded value:



In recent versions of Burp, we can also use the **Burp Inspector** tool to perform encoding and decoding (among other things), which can be found in various places like **Burp Proxy** or **Burp Repeater**:



In ZAP, we can use the **Encoder/Decoder/Hash** tool, which will automatically decode strings using various decoders in the **Decode** tab:



Tip: We can create customized tabs in ZAP's Encoder/Decoder/Hash with the "Add New Tab" button, and then we can add any type of

[Cheat Sheet](#)[Go to Questions](#)

Table of Contents

Getting Started

[Intro to Web Proxies](#)[Setting Up](#)

Web Proxy

[Proxy Setup](#)[Intercepting Web Requests](#)[Intercepting Responses](#)[Automatic Modification](#)[Repeating Requests](#)[Encoding/Decoding](#)[Proxying Tools](#)

Web Fuzzer

[Burp Intruder](#)[ZAP Fuzzer](#)

Web Scanner

[Burp Scanner](#)[ZAP Scanner](#)[Extensions](#)

Skills Assessment

[Skills Assessment - Using Web Proxies](#)

My Workstation

OFFLINE

[Start Instance](#)

00 / 1 spawns left

encoder/decoder we want the text to be shown in. Try to create your own tab with a few encoders/decoders.

Encoding 1

As we can see, the text holds the value `{"username":"guest", "is_admin":false}`. So, if we were performing a penetration test on a web application and find that the cookie holds this value, we may want to test modifying it to see whether it changes our user privileges. So, we can copy the above value, change `guest` to `admin` and `false` to `true`, and try to encode it again using its original encoding method (`base64`):

Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Compass
Logger
Extender
Project options
User options

```

{"username":"admin", "is_admin":true}

```

```

eyJlc2VybmFkZS86ImFkbWluIiwgImIzX2FkbWluIj0cnVlQ==

```

☒ Text
☐ Hex

Decode as ...
Encode as ...
Hash ...
Smart decode

☒ Text
☐ Hex

Decode as ...
Encode as ...
Hash ...
Smart decode

Text to be encoded/decoded/hashed:

```

{"username":"admin", "is_admin":true}

```

Encode
Decode
Hash
Illegal
UTF8
Unicode

Base64 Encode


```

eyJlc2VybmFkZS86ImFkbWluIiwgImIzX2FkbWluIj0cnVlQ==

```

Tip: Burp Decoder output can be directly encoded/decoded with a different encoder. Select the new encoder method in the output pane at the bottom, and it will be encoded/decoded again. In ZAP, we can copy the output text and paste it in the input field above.

We can then copy the base64 encoded string and use it with our request in Burp **Repeater** or ZAP **Request Editor**. The same concept can be used to encode and decode various types of encoded text to perform effective web penetration testing without utilizing other tools to do the encoding.




Connect to Pwnbox

Your own web-based Parrot Linux Instance to play our labs.

Pwnbox Location

UK

13mins

 Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

+1 The string found in the attached file has been encoded several times with various encoders. Try to use the decoding tools we discussed to decode it and get the flag.

```
HTB(3nc0d1n6_n1nj4)
```

Submit

encoded_flag.zip

Hint

Previous

Next

Mark Complete & Next

