Page 9 / Decoding

Decoding

After doing the exercise in the previous section, we got a strange block of text that seems to be encoded:



techniques can further obfuscate the code and make it less readable by humans and less detectable by systems. For that reason, you will very often find obfuscated code containing encoded text blocks that get decoded upon execution. We will cover 3 of the most commonly used text

Base64

base64 encoding is usually used to reduce the use of special characters, as any characters encoded in base64 would be represented in alphanumeric characters, in addition to + and / only. Regardless of the input, even if it is in binary format, the resulting base64 encoded string would only use them.

Spotting Base64

base64 encoded strings are easily spotted since they only contain alpha-numeric characters. However, the most distinctive feature of base64 is its padding using = characters. The length of base64 encoded strings has to be in a multiple of 4. If the resulting output is only 3 characters long, for example, an extra = is added as padding, and so on.

Base64 Encode



Base64 Decode

If we want to decode any base64 encoded string, we can use base64 -d, as follows:

Hex

Spotting Hex

Any string encoded in hex would be comprised of hex characters only, which are 16 characters only: 0-9 and a-f. That makes spotting hex encoded strings just as easy as spotting base64 encoded strings.

Hex Encode

To encode any string into hex in Linux, we can use the xxd -p command:



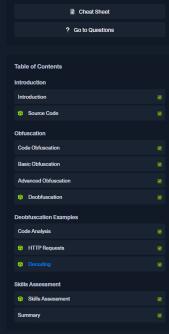
Hex Decode

To decode a hex encoded string, we can use the xxd -p -r command:



Caesar/Rot13

Another common -and very old- encoding technique is a Caesar cipher, which shifts each letter by a fixed number. For example, shifting by 1 $character\ makes\ a\ become\ b, and\ b\ becomes\ c, and\ so\ on.\ Many\ variations\ of\ the\ Caesar\ cipher\ use\ a\ different\ number\ of\ shifts,\ the\ most$





common of which is rot13, which shifts each character 13 times forward.

Spotting Caesar/Rot13

Even though this encoding method makes any text looks random, it is still possible to spot it because each character is mapped to a specific character. For example, in rot13, http://www becomes uggc://jjj, which still holds some resemblances and may be recognized as such.

Rot13 Encode

There isn't a specific command in Linux to do rot13 encoding. However, it is fairly easy to create our own command to do the character shifting:

```
Decoding

MisaelMacias@htb[/htb]$ echo https://www.hackthebox.eu/ | tr 'A-Za-z' 'N-ZA-Mn-za-m'

uggcf://jjj.unpxgurobk.rh/
```

Rot13 Decode

We can use the same previous command to decode rot13 as well:

```
Decoding

MisaetMacias@htb[/htb]$ echo uggcf://jjj.unpxgurobk.rh/ | tr 'A-Za-z' 'N-ZA-Mn-za-m'

https://www.hackthebox.eu/
```

Another option to encode/decode rot13 would be using an online tool, like rot13.

Other Types of Encoding

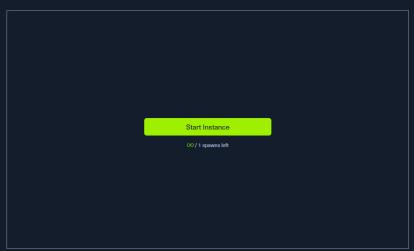
There are hundreds of other encoding methods we can find online. Even though these are the most common, sometimes we will come across other encoding methods, which may require some experience to identify and decode.

If you face any similar types of encoding, first try to determine the type of encoding, and then look for online tools to decode it

Some tools can help us automatically determine the type of encoding, like Cipher Identifier. Try the encoded strings above with Cipher Identifier to see if it can correctly identify the encoding method

Other than encoding, many obfuscation tools utilize encryption, which is encoding a string using a key, which may make the obfuscated code very difficult to reverse engineer and deobfuscate, especially if the decryption key is not stored within the script itself.





Waiting to start...



