

Decoding

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

● ● ●

```
MichaelLuka@htb[/htb]$ curl http://SERVER_IP:PORT/serial.php -X POST -d "param1=sample"

ZG8gdGh1IGV4ZXJjaXN1LCBkb24ndCBjb3B5IGFuZCBwYXN0ZSA7KQo=
```

This is another important aspect of obfuscation that we referred to in **More Obfuscation** in the **Advanced Obfuscation** section. Many 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 encoding methods:

- **base64**
- **hex**
- **rot13**

Base64

base64 encoding is usually used to reduce the use of special characters, as any characters encoded in **base64** would be represented in alpha-numeric 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

To encode any text into **base64** in Linux, we can echo it and pipe it with **|** to **base64**:

● ● ● Base64 Encode

```
MichaelLuka@htb[/htb]$ echo https://www.hackthebox.eu/ | base64

aHR0cHM6Ly93d3cuaGFja3RoZWJveC5ldS8K
```

Base64 Decode

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

● ● ● Base64 Decode

```
MichaelLuka@htb[/htb]$ echo aHR0cHM6Ly93d3cuaGFja3RoZWJveC5ldS8K | base64 -d

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

Hex

Another common encoding method is **hex** encoding, which encodes each character into its **hex** order in the **ASCII** table. For example, **a** is **61** in hex, **b** is **62**, **c** is **63**, and so on. You can find the full **ASCII** table in Linux using the **man ascii** command.

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 Encode

```
MichaelLuka@htb[/htb]$ echo https://www.hackthebox.eu/ | xxd -p

68747470733a2f2f7777772e6861636b746865626f782e65752f0a
```

Hex Decode

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

Hex Decode

```
MichaelLuka@htb[/htb]$ echo 68747470733a2f2f7777772e6861636b746865626f782e65752f0a | xxd -p -r

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

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:

Rot13 Encode

```
MichaelLuka@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:

Rot13 Decode

```
MichaelLuka@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.


Start Instance

1 / 1 spawns left


Waiting to start...

Questions

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

 Cheat Sheet

Target: [Click here to spawn the target system!](#)

+ 1

Using what you learned in this section, determine the type of encoding used in the string you got at previous exercise, and decode it. To get the flag, you can send a 'POST' request to 'serial.php', and set the data as "serial=YOUR_DECODED_OUTPUT".


Submit your answer here...

 Submit

 Hint

 Previous

Next 


 Cheat Sheet

 Go to Questions


Table of Contents


Introduction

Introduction 


 Source Code 

Obfuscation


Code Obfuscation 


Basic Obfuscation 

Advanced Obfuscation 

 Deobfuscation 


Deobfuscation Examples

Code Analysis 

 HTTP Requests 

 [Decoding](#)


Skills Assessment

 Skills Assessment

Summary

My Workstation

OFFLINE

 Start Instance

1 / 1 spawns left