LACUNA SOFTWARE

One-Time Pad Challenge

Microsoft Partner
Gold Application Development
Silver Cloud Platform

One-Time Pad

One-Time Pad (OTP) is a simple, yet unbreakable, encryption technique that consists in combining each character of the plaintext with each character of a secret key, which should be randomly generated and shared between the two parties.

The combination between characters of the key and the message might take place in different ways, the most common being applying a bit-wise XOR, or applying a modular addition, for each character.

One-Time Pad

So far, it looks pretty much like Vigenère Cipher, but it has some peculiarities that are responsible for making it unbreakable:

- The length of the key should be the same as the message's;
- The key should only be used once, and then destroyed.

Challenge

In order to demonstrate that the misuse of cryptography can lead into major security flaws, this challenge will consist of a server that uses One-Time Pad cipher (applying bit-wise XOR for each character) to encrypt and decrypt its authentication tokens, **but always reuses the secret key.**

Your objective, as challenger, is to forge a valid token for the user “master” — by exploiting the server’s misuse of OTP cipher, mentioned above — and then access an endpoint which only the master has access to.

Challenge – The Token

- When issuing the token, the server will encrypt it using OTP cipher, applying bit-wise XOR for each character of the plaintext and the key, and **will use the same key for every token that is issued.**
- Inside the encrypted token, one of the information that is encoded is the username. For example, if the username of a user that logged in is “testuser”, then the token, before being encrypted, will contain the string “testuser”.

Challenge – The Token

- The server will only issue and receive the token in hex format (ex: 12AB8E…)
- Inside the token, the max. length of a username is 32. If the username is shorter than this, a **fixed** padding character will be used to fill the remaining space inside the token.
- Each token expires in 20 seconds.

Challenge – API

- The URL of the server that will be used in this challenge is “**weak-system-lab.lacunasoftware.com**”

  Simple site response: https://weak-system-lab.lacunasoftware.com/api/system/info

- The server will return 200 for **all** the requests that it can understand/process, regardless of whether it has been successful or not. The difference lays in the response:
  - Example of error:

```
{
        “status”: “Fail”,
        “message”: “Fail reason…”
}
```

Challenge – API

- Example of Success:

```
{
        “status”: “Success”
}
```

Challenge – API

[POST] /api/users/create

- Endpoint used to register a user.
- It expects, as body, a JSON with the fields “username”, “password” and “email”, as in the following example:

```
{
    "username": "testtest1",
    "password": "testtest",
    "email": "test1@test.com"
}
```

- The username must be between 8 and 32 alphanumeric characters, and the password needs to be at least 8 characters. Remember to pass you real email in the field “email” (we will use it to know if you passed the test or not).
- It returns 200, and an object with the field “status” set to “Success”, or “Fail”, as exemplified previously.

Challenge – API

[POST] /api/users/login

- Endpoint used to emit a new token for a given user.
- It expects, as body, a JSON with the fields “username” and “password”, as in the following example:

```
{
    "username": "testtest1",
    "password": "testtest"
}
```

- It returns 200, and an object with the fields “token” and “response. Example:

```
{
  “status”: “Success”,
  “token”: “AEFB12398…”
}
```

Challenge – API

[GET] /api/secret

- Endpoint whose access is limited to the user “master”. Your goal is to access this API, by forging a token with the username “master”.
- As due authorization is necessary for this endpoint, you need to pass a valid token in the Authorization Header, using the prefix “Bearer ” (bearer scheme). Example:

KEY	VALUE
Authorization	Bearer 582469702E0889210B544...

- It returns 200, and an object with the fields “status” and “secret” with a secret message in case you successfully forged a master user token. Example:

```
{
	“status”: “Success”,
	“secret”: “##########################”
}
```

Challenge – Hints

- Important XOR property: *self-inverse* ($A \oplus A = 0$)
  - $A \oplus B = C \Rightarrow C \oplus B = A$
- You can request a new token as many times as you want.
- You can create as many users as you want to, as long as you use your real email (as we are going to use that info to know if you passed the test or not).

Challenge – Thoughts

- How could we improve the algorithm, or use another algorithm, in order to prevent token tampering
- Any other

THANK YOU, AND GOOD LUCK!