This challenge is ‘P.A.S trop dur’

The goal of this challenge was to retrieve the password that the user used to use the webshell that was already on the server. After looking at the pcap, we notice the “maliciouswebshell.php”, which sounds fishy.

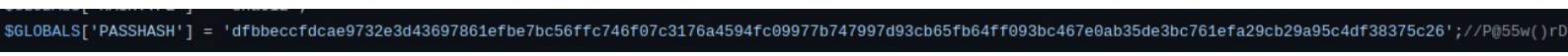
```
 466 GET /assets/maliciouswebshell.php HTTP/1.1
 903 HTTP/1.1 200 OK  (text/html)
 778 GET /assets/maliciouswebshell.php HTTP/1.1
1575 HTTP/1.1 200 OK  (text/html)
 790 GET /assets/maliciouswebshell.php HTTP/1.1
2497 HTTP/1.1 200 OK  (text/html)
 792 GET /assets/maliciouswebshell.php HTTP/1.1
4928 HTTP/1.1 200 OK  (text/html)
 695 GET /assets/maliciouswebshell.php HTTP/1.1
```

fishy, innit ?

After looking closely at the data exchanged with a said php file, and with the help of a quick internet search, we find that it is the P.A.S. webshell (https://github.com/cr1f/P.A.S.-Fork).

Quite a neat webshell

First, we try the password that is already implemented in the file, [P@55w()rD], but it does not work, the attacker may have changed the default password (I did ^^).



the default password

Looking at the first exchanged packets, we notice that no POST request is sent. Instead, the webshell is using cookies. The transition between the form and the cookie is done through javascript.

```
function submitViaCookie(encodedForm, refresh = true){
        var reqlen = 0;
        var elements = encodedForm.getElementsByTagName("*");
```

Now, we just need to understand what makes the cookies.
We always have two pairs of name=value.

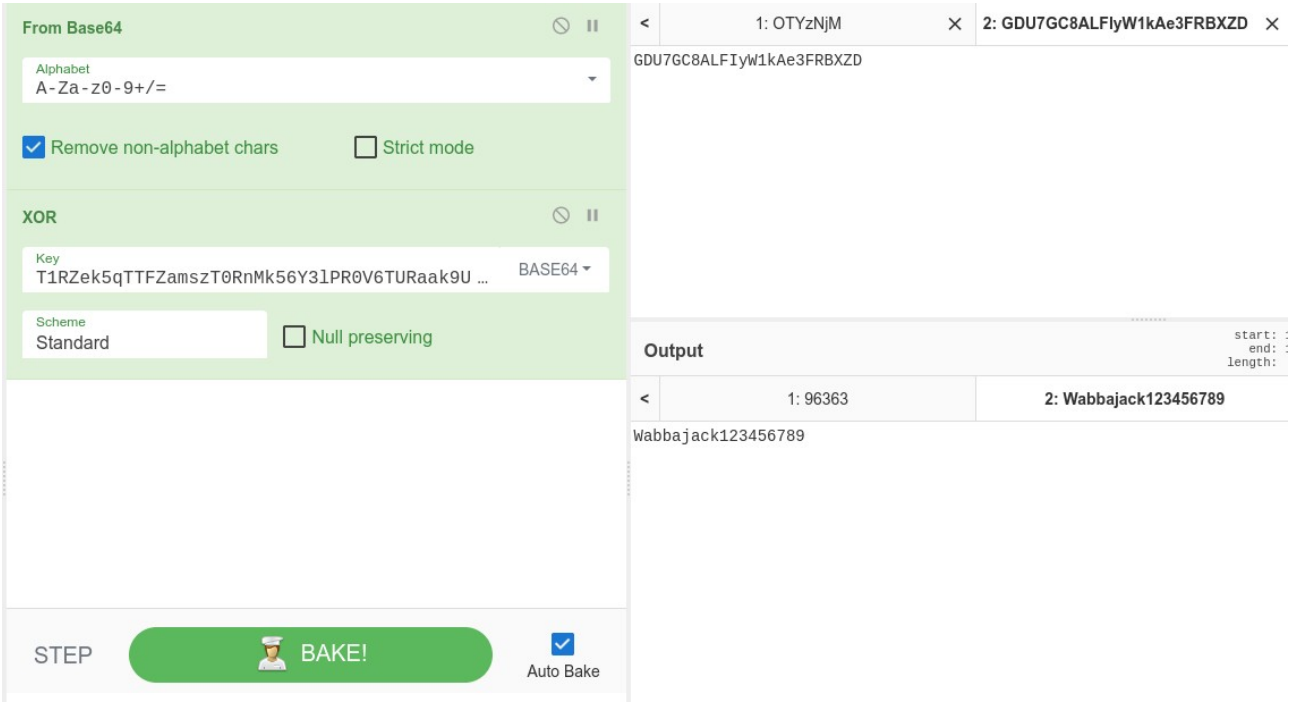
```
Cookie pair: OTYzNjM3f352a09=GDU7GC8ALFIyW1kAe3FRBXZD
Cookie pair [truncated]: MjNzYTOOTYzNjM=T1RZek5qTTFZamszT0RnMk56Y3lPR0V6TURaak9UTXpPRGxqTVl
```

The PHP file generates an encoding key ENCKEY based on the hardcoded password's hash along with a number PRELEN that is used to truncate the encoding key in some parts.

The first name (OTYzNjM3f352a09) is the concatenation of ENCKEY[:PRELEN] with ENCKEY[:PRELEN] xored with the form.name (here it is "pass") in hexadecimal.
The first value (GDU...) is the base64 (user's input xor ENCKEY) .

The second name is the concatenation of reverse(ENCKEY[:PRELEN]) with ENCKEY[:PRELEN]
The second value (T1RZ...) is the base64 of the encoding key.

In the end, it was just required to use cyberchef to xor the ENCKEY with the base64(input) that was in the capture and voilà !



Please note that the real flag was Wabbajack12345678 as there were some wrong inputs when i took the capture

FLAG: HACKDAY{Wabbajack12345678}

Written by Chelinka. March 20th 2023