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

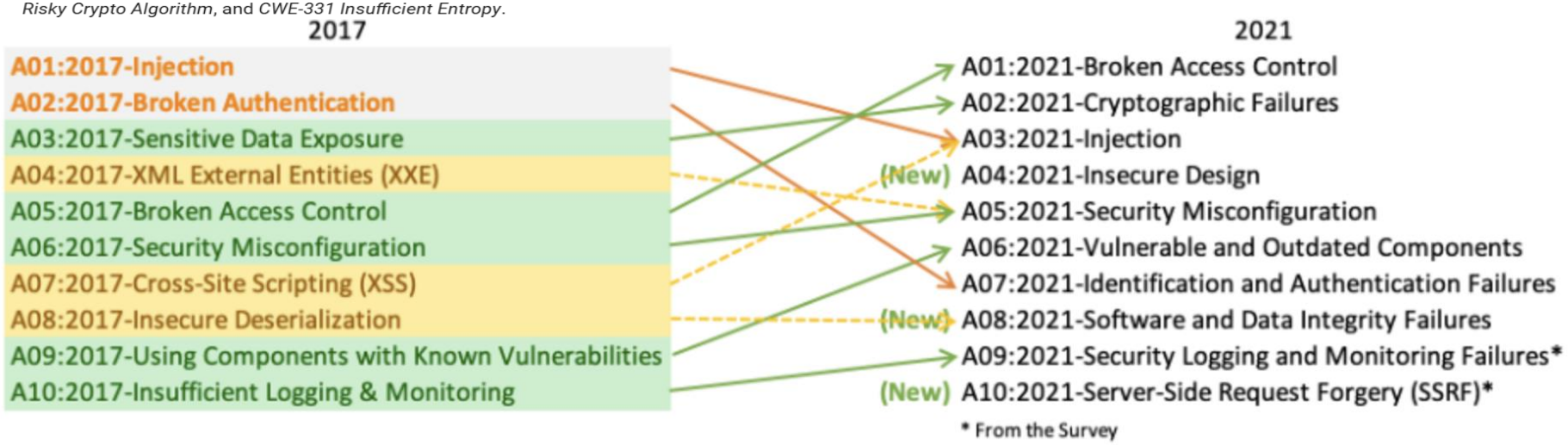
A02:2021 – Cryptographic Failures

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*.



#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

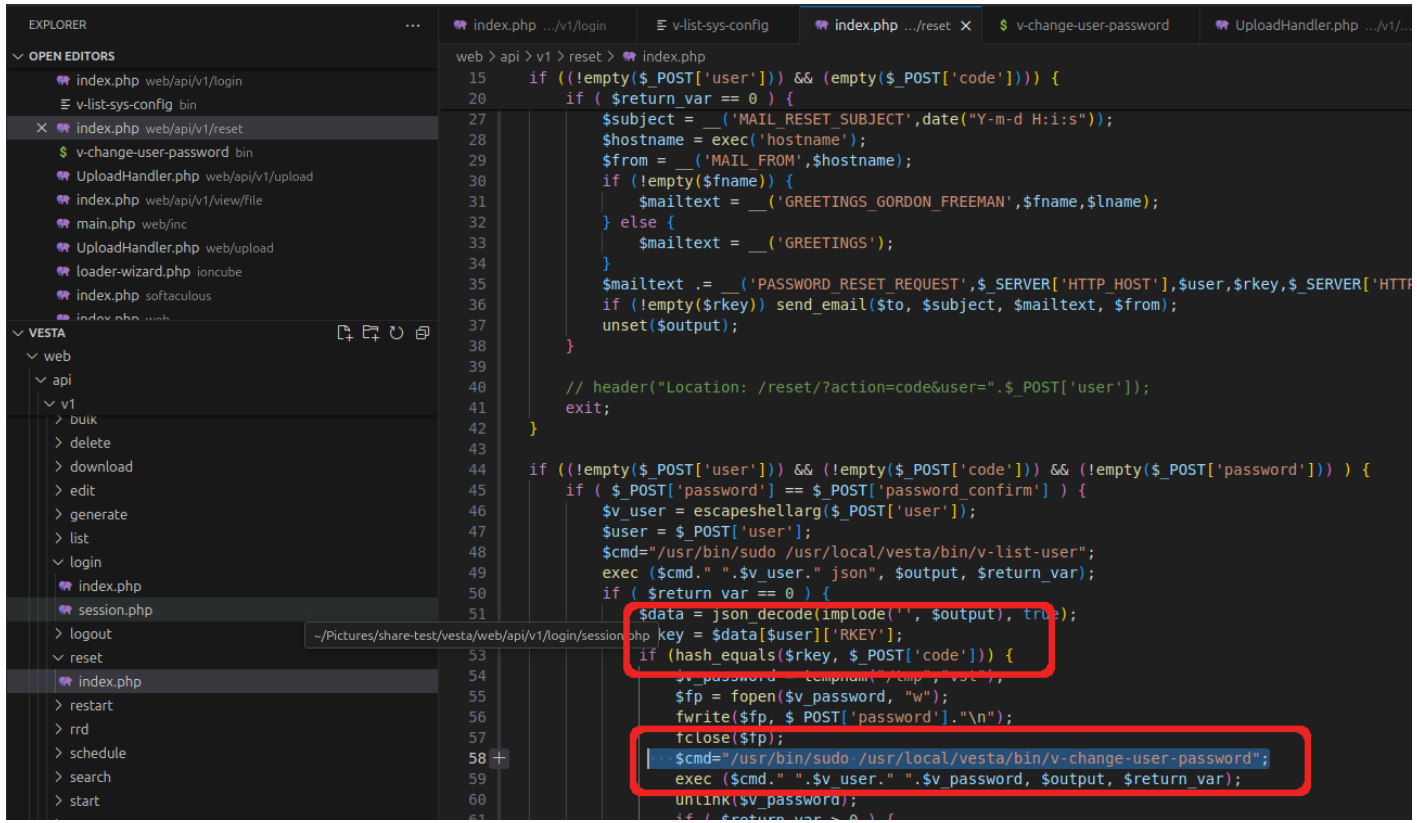
VESTA CP – White Box Pentest

- 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

Standard password reset email

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



PHP API calls bash for password reset

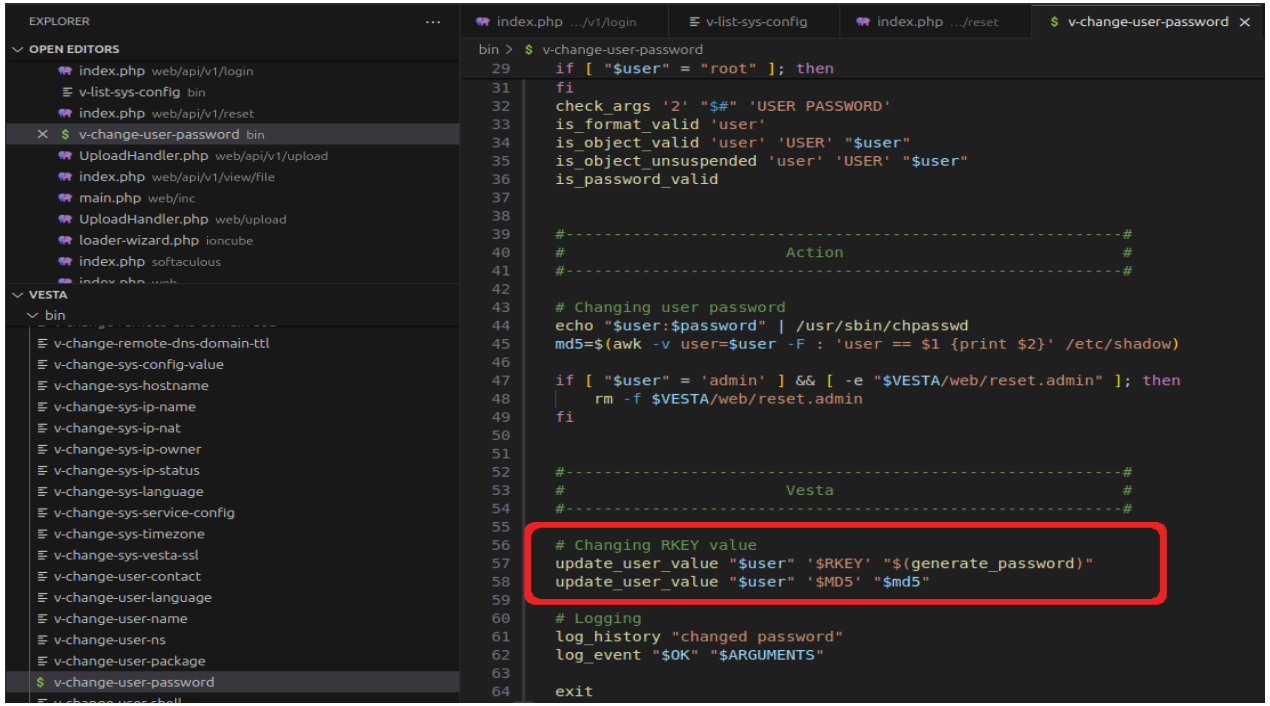
Vesta CP – What is RKEY ?

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

RKEY is the password reset token

Vesta CP – v-change-user-password script

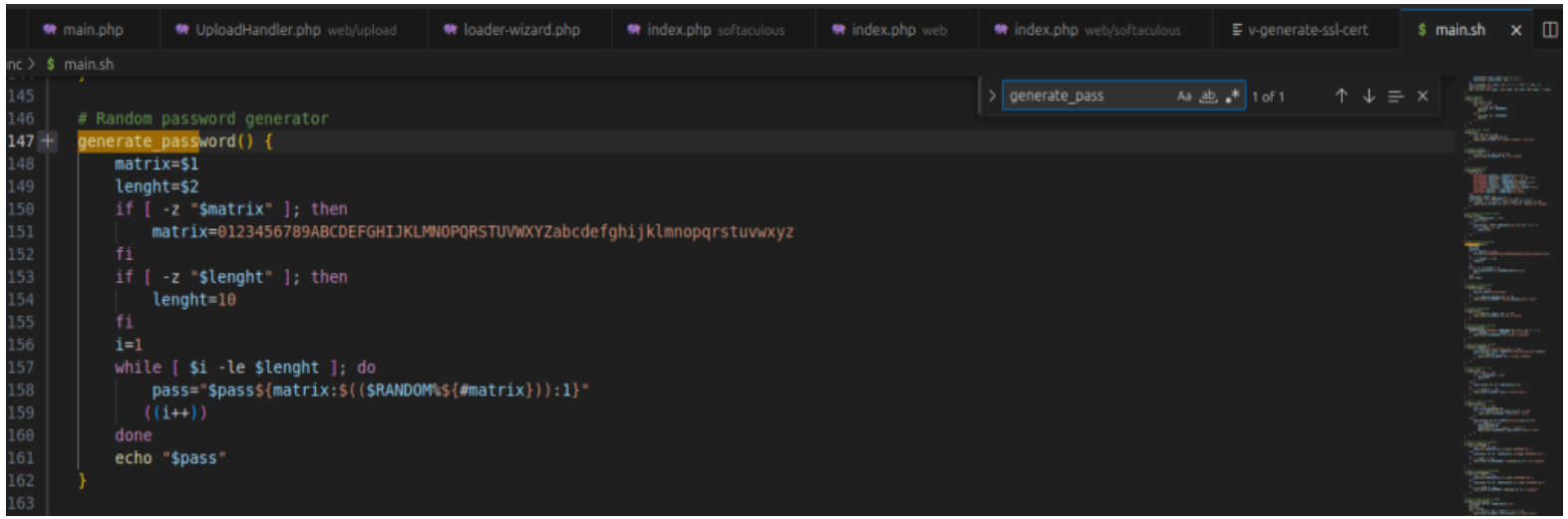
- PHP API calls this bash script
- It changes the password AND
- It generates the RKEY for the NEXT password reset



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



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

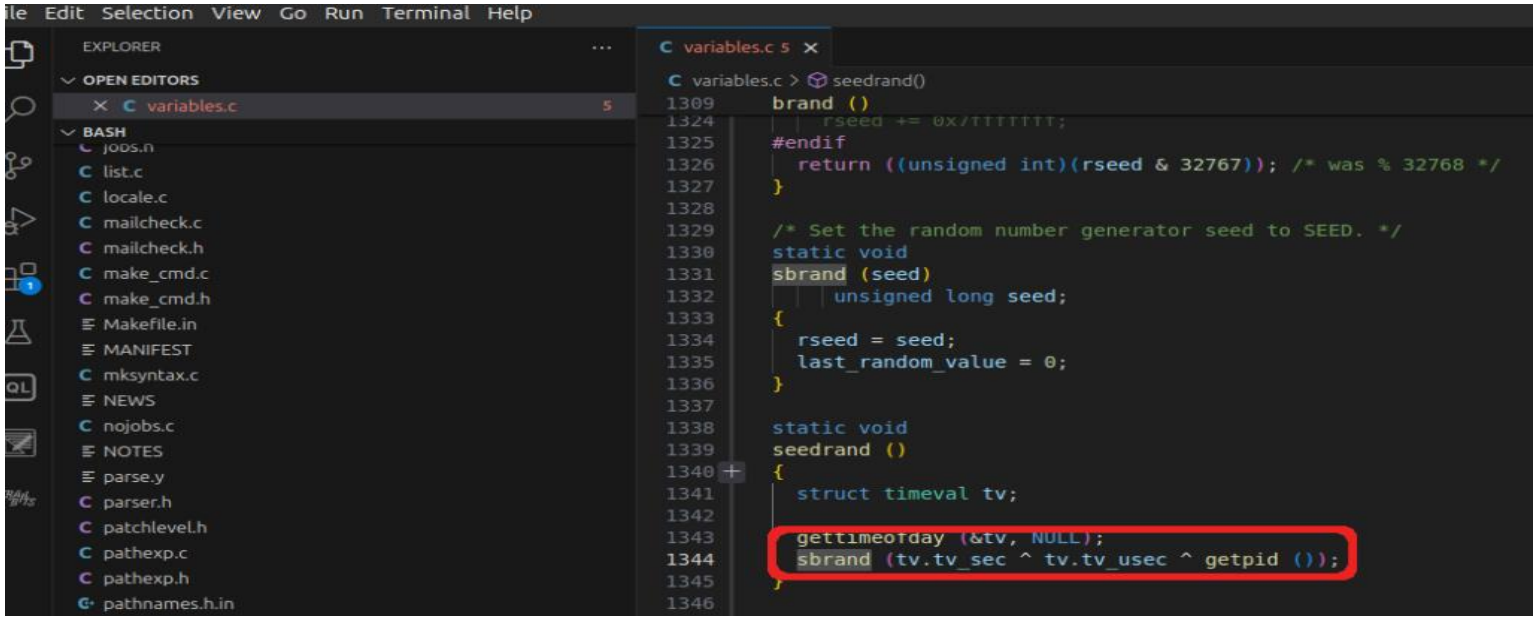
$ RANDOM=1337; echo $RANDOM $RANDOM $RANDOM
24879 21848 15683
$ RANDOM=1337; echo $RANDOM $RANDOM $RANDOM
24879 21848 15683

https://github.com/jorianwoltjer/bashrandomcracker

Vesta CP – Quick Dive into bash internals (C)

It seeds the generator with

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



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)

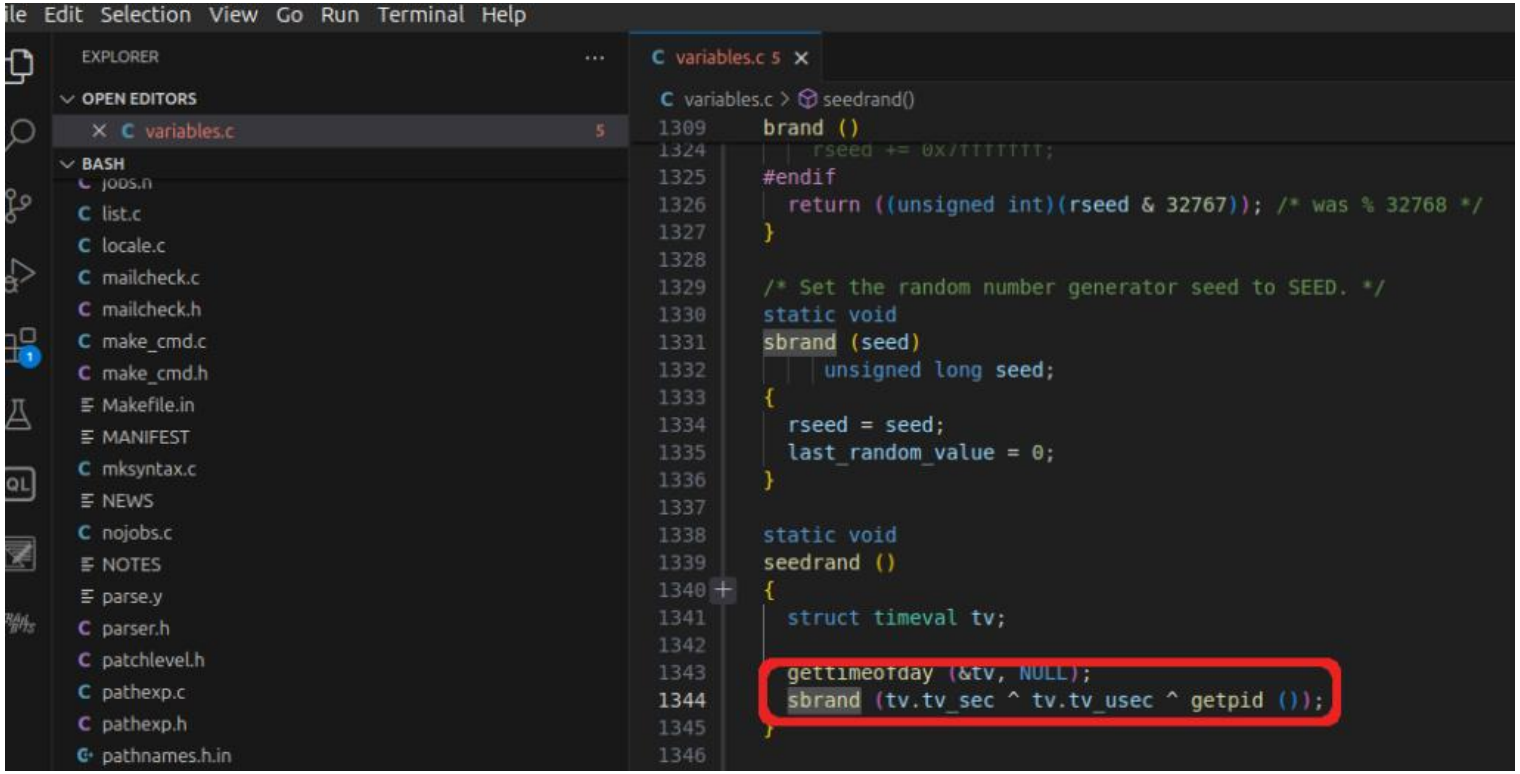
LCG function uses s1 and s2

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

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.

- 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 brutefoce 4B+
- We suggest to bruteforce only the timestamp for the past 1–3 years

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
```

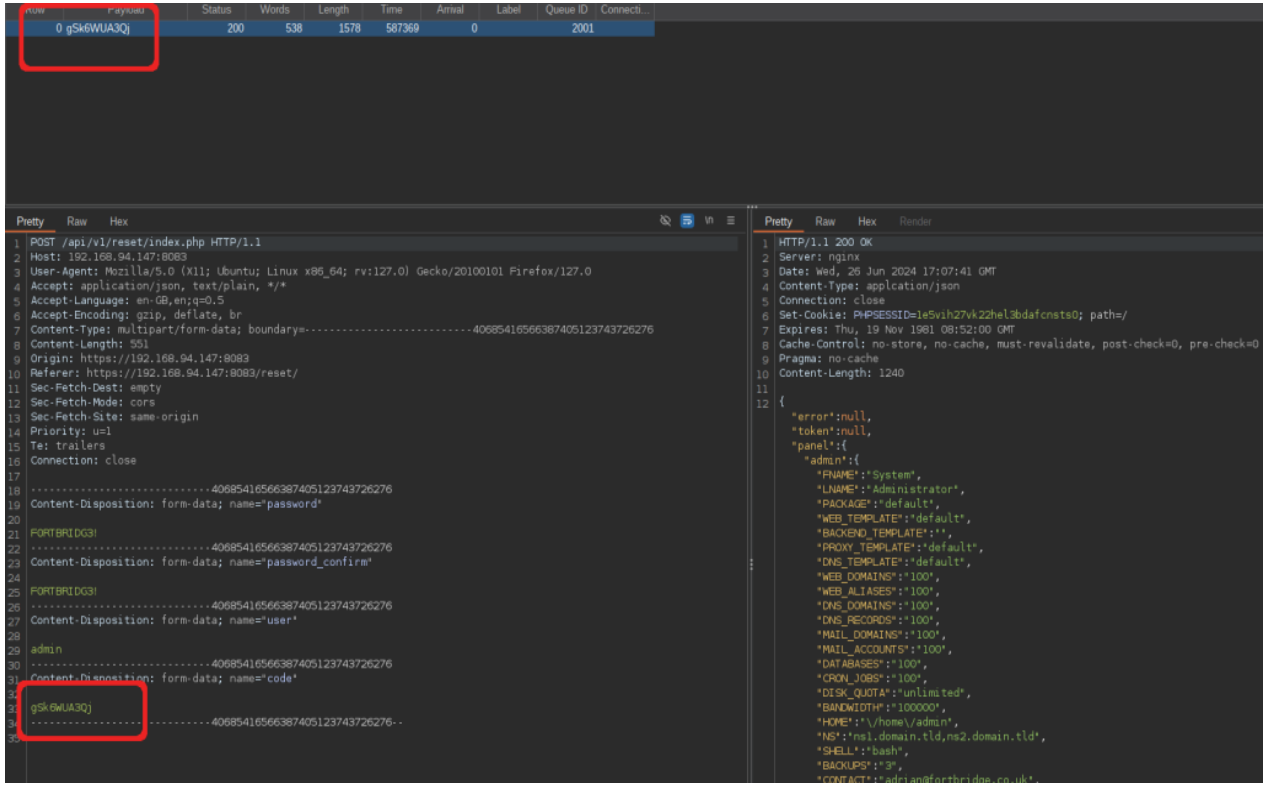
```
root@ubuntu:/usr/local/vesta# cat data/users/admin/user.conf
FNAME='System'
LNAME='Administrator'
PACKAGE='default'
WEB_TEMPLATE='default'
PROXY_TEMPLATE='default'
DNS_TEMPLATE='default'
WEB_DOMAINS='100'
WEB_ALIASES='100'
DNS_DOMAINS='100'
DNS_RECORDS='100'
MAIL_DOMAINS='100'
MAIL_ACCOUNTS='100'
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'
...FlM8iHdzgUB1xxC0kPUNyQSFA8eW1DEDmPr06bKWxfhsELB4nSlYz3SmFFieD6UMD33McaCdupKIAtkKFozxL0'
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



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