



# Artemis: How CERT PL finds vulnerabilities in our constituency at a scale

Krzysztof Zajac

cert.pl

# whoami

- Krzysztof Zajac
- Senior Threat Analysis Specialist, CERT PL
- Teaches offensive security at the University of Warsaw

# The purpose of this talk

# 1. Show the approach

I'll describe the non-technical environment as well

---

## 2. Show the tool

---

# 3. Encourage you to start similar projects

(if you are in the position to do so)

---

All of these are equally important!














# Purpose

After an incident, let's make sure it won't occur in other entities.

Example:

- exposed code repository on an university website caused API key leak and unauthorized data access
- let's check whether other entities have exposed code repositories!

## Index of /.git

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>			-
 <a href="#">FETCH_HEAD</a>	2019-04-03 12:19	3.8K	
 <a href="#">HEAD</a>	2019-04-03 15:04	23	
 <a href="#">ORIG_HEAD</a>	2019-04-03 12:18	41	
 <a href="#">config</a>	2019-04-03 12:18	312	
 <a href="#">description</a>	2019-04-03 12:19	73	
 <a href="#">hooks/</a>	2019-04-03 12:19	-	
 <a href="#">index</a>	2019-04-03 15:04	226K	
 <a href="#">info/</a>	2019-04-03 12:19	-	
 <a href="#">logs/</a>	2019-04-03 12:19	-	
 <a href="#">objects/</a>	2019-04-03 12:19	-	
 <a href="#">packed-refs</a>	2019-04-03 12:19	22K	
 <a href="#">refs/</a>	2019-04-03 12:19	-	



A domain →  **artemis** →

1. The following addresses contain version control system data:

- [https://\[REDACTED\]:443/.git/](https://[REDACTED]:443/.git/)

(...)



1. The following addresses contain version control system data:

- [https://\[REDACTED\]:443/.git/](https://[REDACTED]:443/.git/)

(...)

2. The following addresses contain old Joomla versions:

- [https://\[REDACTED\]:443](https://[REDACTED]:443) - Joomla 2.5.4

(...)

.gov.pl,  
schools,  
hospitals,  
universities,  
banks,  
...



**artemis**



1. The following addresses contain version control system data:

- [https://\[REDACTED\]:443/.git/](https://[REDACTED]:443/.git/)
- (...)

2. The following addresses contain old Joomla versions:

- [https://\[REDACTED\]:443](https://[REDACTED]:443) - Joomla 2.5.4
- (...)

What do we check?

---

# A couple dozen modules

- Finding subdomains (e.g. cert.pl → test.cert.pl)
- Domain expiration check
- Bad DNS configuration check:
  - Zone transfer
  - Subdomain takeover
- E-mail spoof protection mechanisms: SPF/DMARC
- Bad/expired TLS certificates, https:// redirect
- Port scanning, identifying services on a given server

# A couple dozen modules

- WordPress, WordPress plugin, Drupal, and Joomla version check
- Closed WordPress plugins
- **Nuclei support:** thousands of vulnerabilities and misconfigurations (from Open Redirects to RCEs)
- SQLi and XSS check
- Scripts loaded from nonexistent domains

# A couple dozen modules

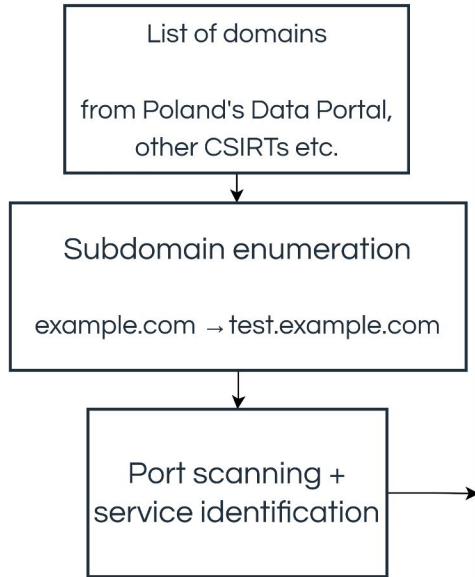
- Directory index
- Weak passwords
- Exposed code repositories
- Exposed login panels, remote desktop, databases, ...
- Accidentally published files (eg. /db.sql, /backup.zip or /wp-config.php.bak)
- **Possibility to integrate any other tool (commercial or open-source) + an example how to do that**

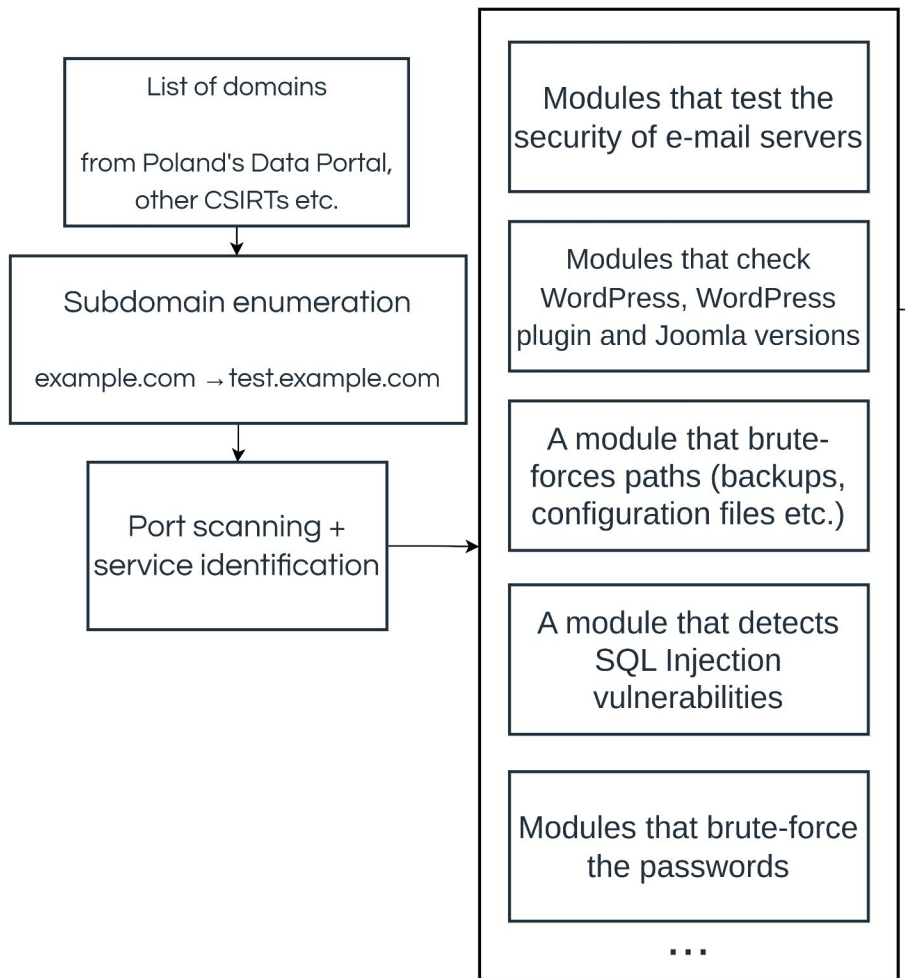
# Where the list comes from?

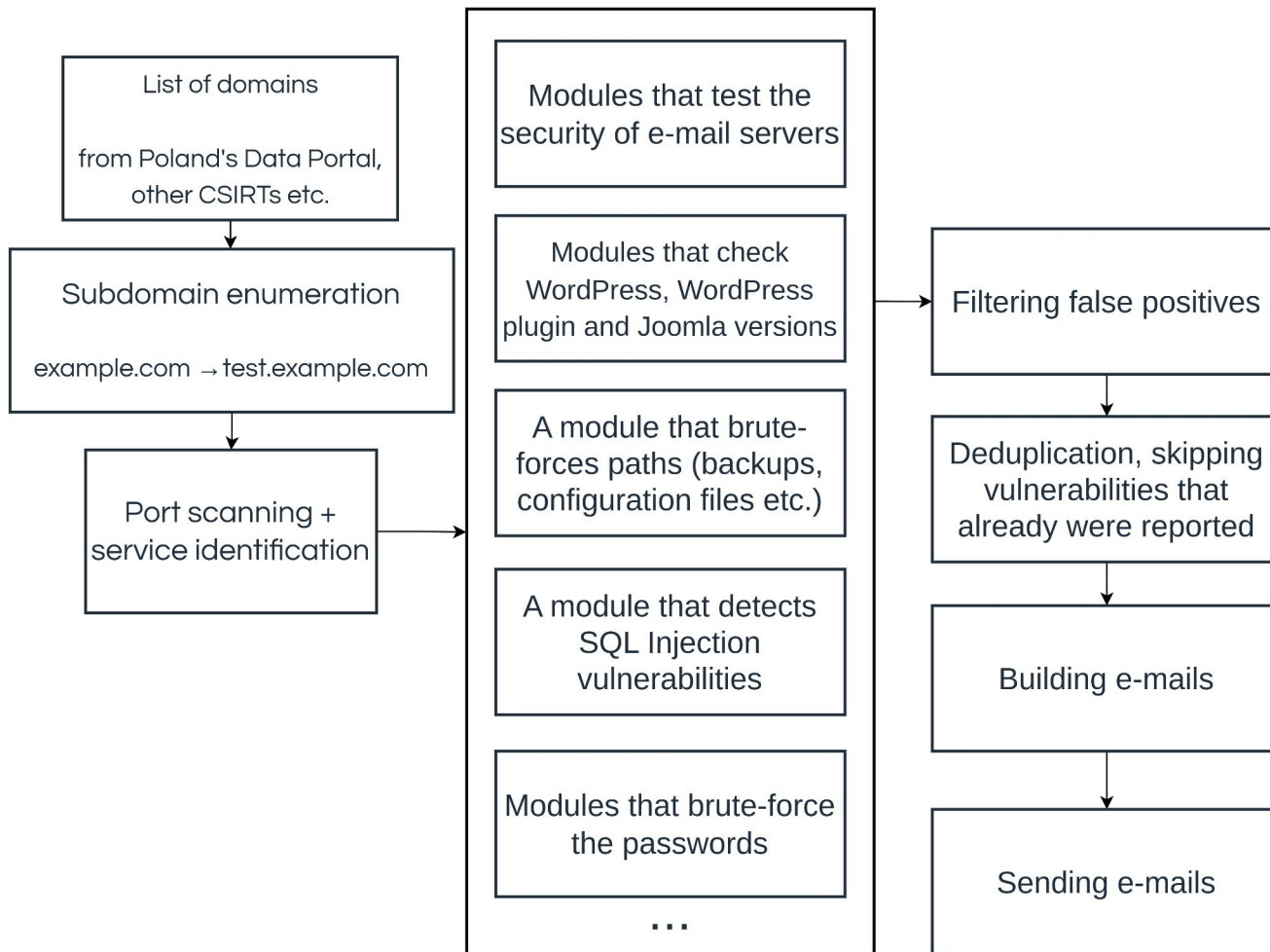
Our experiences in handling incidents.

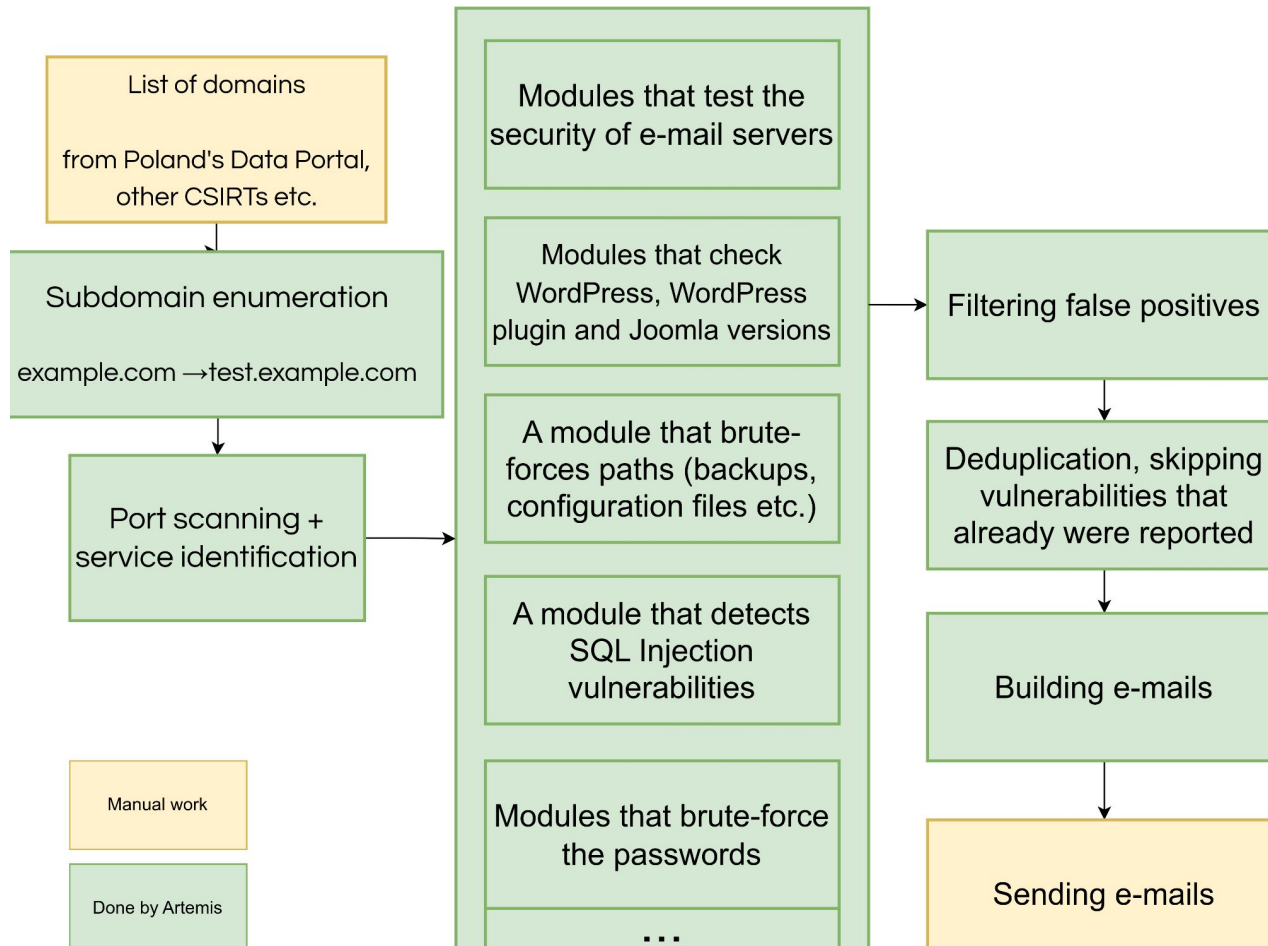
E.g.: someone got hacked because of a SQL Injection vulnerability? Let's improve SQL Injection detection capabilities!











# Example e-mail

The following addresses contain version control system data:

- [https://\[REDACTED\]:443/.git/](https://[REDACTED]:443/.git/)

Making a code repository public may allow an attacker to learn the inner workings of a system, and if it contains passwords or API keys - also gain unauthorized access. Such data shouldn't be publicly available.

# Example e-mail

**Such reports are sent by CERT PL to scanned entities  
(but in Polish).**

# List of domains

- Customer dataase (if you're e.g. a hosting provider)
- Data portals: <https://dane.gov.pl/en>
- Tools such as crt.sh: <https://crt.sh/?q=%25.gov.de>,
- Custom databases (example: [rspo.gov.pl](https://rspo.gov.pl) for schools),
- Be creative (example: [mamprawowiedziec.pl](https://mamprawowiedziec.pl)),
- Contacting the entities (slow)
- ...

# List of domains

- ~~— Customer database (if you're e.g. a hosting provider)~~
- **Data portals: <https://dane.gov.pl/en>**
- **Tools such as crt.sh: <https://crt.sh/?q=%25.gov.de>,**
- **Custom databases (example: [rspo.gov.pl](https://rspo.gov.pl) for schools),**
- **Be creative (example: [mamprawowiedziec.pl](https://mamprawowiedziec.pl)),**
- **Contacting the entities (slow)**
- ...



# Who do we scan

- All gov.pl domains
- Local government entities
- Municipal corporations: water management, waste collection, ...
- Key Service Operators
- Banks
- Universities, schools, preschools and other educational entities
- Hospitals

# Who do we scan

- Local and country-level newspapers, TVs, information portals, etc.
- Websites of politicians, political parties, candidates, etc.
- Professional self-governments (e.g. medical chambers)
- Lists of domains provided e.g. by other CSIRTs or ministries.
- Domains provided voluntarily by companies.

# Legal basis

Don't design law that:

- Allows scanning of a small subset of entities (e.g. *important* ones)
- Requires actions that are **not viable** in case of broad scans, such as signed agreements with entities.

Possibility to perform **broad scans** was **crucial** for the success of Artemis!

(keep an eye on the above when implementing EU NIS2)

# Statistics

**251.6k** domains/IPs and **635.2k** subdomains scanned  
**periodically.**

# Statistics

Since January 2023 we reported ~**448.5k** vulnerabilities and misconfigurations, including:

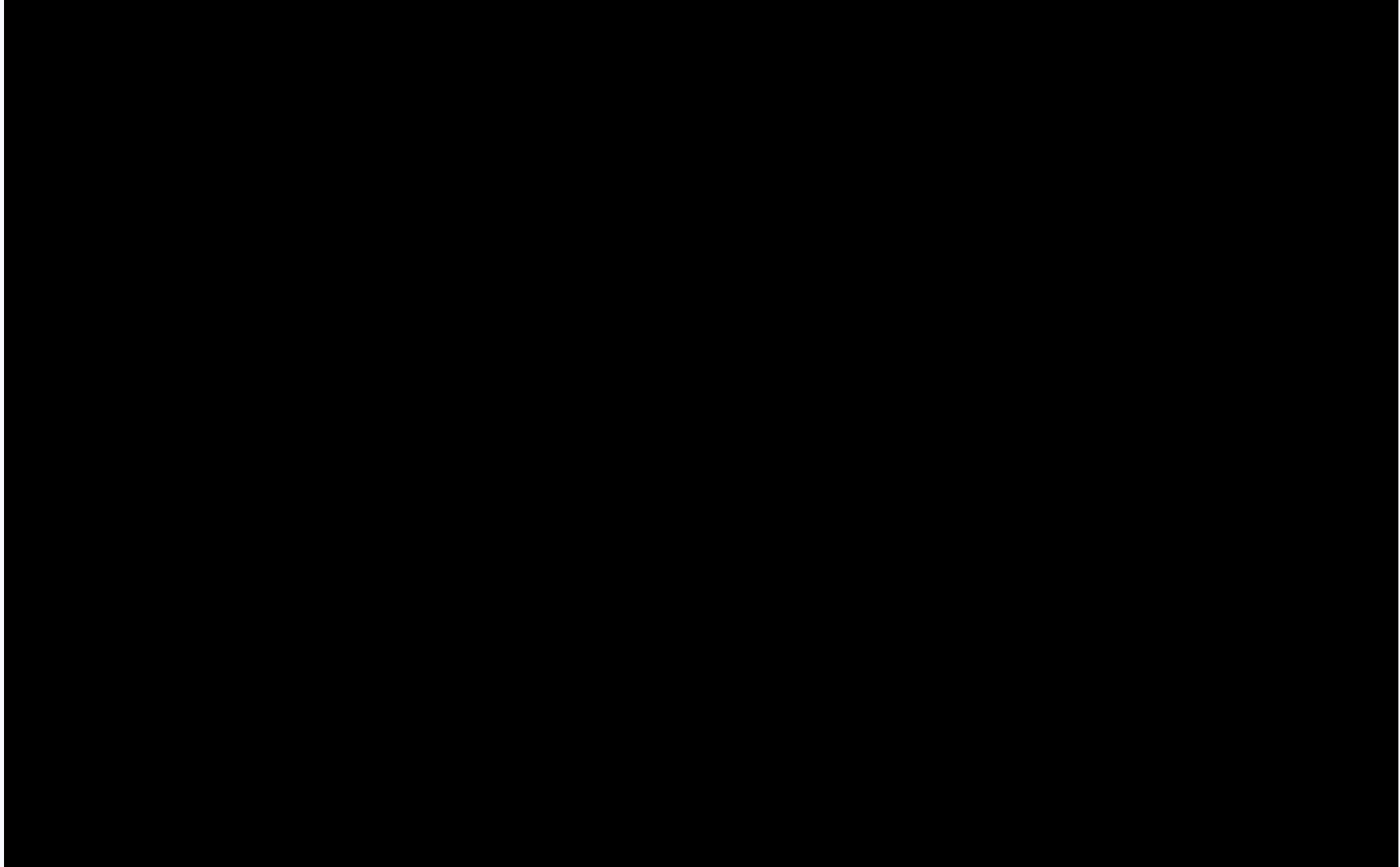
- ~**27.6k** high-severity,
  - ~**284.9k** medium-severity,
  - ~**136k** low-severity.
- For example we have over **1000** confirmed SQL Injections (where we managed to **dump data from the database**).

# Communication

- We already sent over **110k e-mails**.
- If an entity doesn't fix a serious issue, **we call them**. We already made around **6k** such calls.
- Reactions are mostly positive (but we sometimes receive bug reports).
- Important: sometimes our e-mail **gives “political” support to the admins** even if they know about a problem.

# Demo

---





export.zip — Ark

Archive File Settings Help

Extract Preview Open Find... Add Files... Delete

Name	Size	Compressed	Mode	CR
advanced	4 Files		drwxrwxr-x	
messages	1 File		drwxrwxr-x	
entity1.html	943 B	439 B	-rw-r--r--	4D
stats.txt	66 B	58 B	-rw-r--r--	CC

entity1.html  
943 B  
Type: HTML document

Search:

[ks](#) | [Show results](#)

[ks](#) | [Show results](#)

[ks](#) | [Show results](#)

entity1.html

File /tmp/ark-ALitZb/messages/entity1.html

Artemis stats - D... Artemis stats - D... Artemis stats - D...

- The following domains don't have properly configured e-mail sender verification mechanisms:
  - ██████.pl: Valid DMARC record not found. We recommend using all three mechanisms: SPF, DKIM and DMARC to decrease the possibility of successful e-mail message spoofing.

These mechanisms greatly increase the chance that the recipient server will reject a spoofed message. Even if a domain is not used to send e-mails, SPF and DMARC records are needed to reduce the possibility to spoof e-mails.

ous 1 Next

# Exporting the reports



**artemis** →



# Exporting the reports

Export: **the most important feature of Artemis!**

And what with the problems that didn't make it to the reports?

Artemis checks whether a problem is interesting enough to be included in the reports.

API



# Report translations

---

# Translations

Lazy way:

*translate the following file to French, leaving stuff in quotes after "msgid" in English but putting the translation after "msgstr":*

# Our approach

- **Being open:** <https://cert.pl/skanowanie/> (translation [here](#))
- Not overloading the servers (our configuration: one request per second per IP)
- Not waiting for the scan to finish, sending reports every 2 weeks
- Making sure the vulnerabilities get fixed - e.g. **calling the scanned entities**
- Allow submitting domains voluntarily (even in a **low-tech way** - e-mails)

# Artemis in production

Most important: **do things well enough.**

Examples:

- A module is broken? Scan using the rest until it gets fixed.
- Don't yet have a green light to scan all entities? Scan the ones you are allowed to.



# Conclusions

- Lots of low-hanging vulnerabilities
- Many good offensive tools are available  
even plain Nuclei scan or WordPress/Joomla version check would find many vulnerabilities
- Iterative development contributed to the project  
success

Instead of building the best scanner possible, we built a MVP with a subset of modules and ran initial scans. During scans, we observed bugs, fixed them, but also added new modules.

# How to start

- Start small!
- Download Artemis (and <https://github.com/CERT-Polska/Artemis-modules-extra>)
- Set up Artemis using the [quick-start documentation](#)
- Take one list of domains (e.g. one you can get easy approval to scan), e.g. from a [data portal](#)

# How to start

- translate Artemis to your language - we have docs on how to do that.
- Scan, send the results.
- Show to the shareholders that the scanning makes sense.
- Iterate: increase scanning coverage.
- Contact [artemis@cert.pl](mailto:artemis@cert.pl) in case of any problems.

# How to start

**CERT PL will be glad to help with setting up your scanning pipeline.**

Good luck!

---

45



Questions?

# Links

<https://github.com/CERT-Polska/Artemis>

<https://github.com/CERT-Polska/Artemis-modules-extra>

[artemis@cert.pl](mailto:artemis@cert.pl)

<https://discord.com/invite/GfUW4mZmy9>