Artemis: how CERT PL improves the security of the Polish internet

Krzysztof Zając
69th TF-CSIRT Meeting
2023-05-24, Bucharest
Purpose

- Checking the security of websites and systems used by institutions in our constituency:
  - schools
  - universities
  - government institutions
  - local government and public utility companies
  - hospitals
- Improving the security of these systems by reporting the vulnerabilities
Legal basis

Act of National Cybersecurity System Legislation (NIS1 implementation) article 26 - tasks of CSIRTs:

1. monitoring threats and incidents on a country level
2. providing information to entities in the national cybersecurity system
3. in justified cases: conducting vulnerability research of a device or software
4. developing tools to (...) detect and combat cybersecurity threats (...).

Article 32 allows us to do “any necessary technical actions” to analyze cybersecurity threats
Legal basis

- We are the registrar for the .pl domain - we can put a clause that allows us to scan in gov.pl rules.

- In some cases - agreements with other CSIRTs or institutions directly responsible for a system.

- Backup: penal code article 269c. The Polish Criminal Code penalises breaking into someone else’s IT systems, but has an explicit exception for when it’s done for security purposes, without breaking anything and if the issue was immediately reported.

- NIS2 implementation - upcoming.
Design goals

- Low amount of manual vulnerability analysis: heuristics to filter true from false positives
- Low load on scanned systems: per-host rate limiting
- Reusing existing tools
- Scalability
- Easy integration of a new tool
- Flexible scanning pipeline
  - work with domains, HTTP services, WordPress instances, …
What Artemis does?

- Finds subdomains using open-source sources (crt.sh, Common Crawl, Wayback Machine, ...):
  
  example.com → mail.example.com, old.example.com

- Detects DNS misconfigurations:
  - Zone transfer,
  - Subdomain takeover.
What Artemis does?

- Performs port scanning and service identification (is this a website? a database?).

- Finds backups and other interesting files (e.g. /wp-config.php.bak) using brute-force.

- Brute-forces weak passwords (FTP, PostgreSQL, MySQL and WordPress).
What Artemis does?

Detects directory index:

```
Index of /

Name  Last modified  Size  Description
```
What Artemis does?

Detects known vulnerabilities using Nuclei:

CVE-2022-1040, CVE-2022-1020, CVE-2022-1013...
What Artemis does?

- Checks e-mail configuration (SPF, DMARC, open relay).
- Detects SQL Injection vulnerabilities.
- Detects accidentally published VCS repositories.
- Verifies SSL/TLS configuration.
List of domains from Poland's Data Portal, other CSIRTs etc.

Subdomain enumeration
example.com → test.example.com

Port scanning + service identification

Modules that test the security of e-mail servers

Modules that check WordPress, WordPress plugin and Joomla versions

A module that brute-forces paths (backups, configuration files etc.)

A module that detects SQL Injection vulnerabilities

Modules that brute-force the passwords

...
Add targets

Targets (separated with newlines)

Batch file (should contain one target per line)

Choose File No file chosen

Tag

You may provide any string here - it will be saved in the task results in the database so that you can e.g. use the value when processing the results automatically.

Start scan
## Analysis of test.local

### Tasks

<table>
<thead>
<tr>
<th>created at</th>
<th>receiver</th>
<th>task</th>
<th>status: reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2023-02-20 08:47:44</td>
<td>mail_dns_scanner</td>
<td>test.local</td>
<td>Found problems: DMARC record is not present</td>
</tr>
<tr>
<td>2023-02-20 08:47:54</td>
<td>directory_index</td>
<td>test.local:8001</td>
<td>Found directories with index enabled: http/test.local:8001/files/</td>
</tr>
<tr>
<td>2023-02-20 08:47:56</td>
<td>vcs</td>
<td>test.local:8002</td>
<td>Found version control system data: git</td>
</tr>
</tbody>
</table>

Showing 1 to 5 of 5 entries (filtered from 43 total entries)
Example e-mail

1. The following addresses contain version control system data:
   - https://███: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.

2. The following addresses contain old Joomla versions:
   - https://███:443 - Joomla 2.5.4

   If a site is no longer used, we recommend shutting it down to eliminate the risk of exploitation of known vulnerabilities in older Joomla versions. Otherwise, we recommend regular Joomla core and plugin updates.

3. The following domains don’t have properly configured e-mail sender verification mechanisms:
   - □□.pl: Valid SPF record not found
   - □□.pl: Valid DMARC record not found

   Such configuration may allow an attacker to send spoofed e-mail messages from these domains.
Artemis is open-source

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

We invite you to use Artemis and add your own modules!

Not all modules are open-source yet - we are currently open-sourcing the module to build e-mails.
How to write a new module

Let's assume you want to check whether the URL contains the string suspicious.

```python
class CustomScanner(ArtemisBase):
    # Module name that will be displayed
    identity = "custom"
    
    # Types of tasks that will be consumed by the module - here,
    # open ports that were identified as containing a HTTP/HTTPS service.
    filters = [{"type": TaskType.SERVICE, "service": Service.HTTP}]

    def run(self, task: Task) -> None:
        # Will convert the identified service to the form of a URL,
        # e.g. http://domain.com:8001/
        url = get_target_url(task)

        if "suspicious" in url:
            status = TaskStatus.INTERESTING
            reason = "suspicious link detected!"
        else:
            status = TaskStatus.OK
            reason = None

        self.db.save_task_result(task=task, status=status, status_reason=reason)
```
Modules can produce/consume various types of objects

- **classifier**: decides whether a task is a domain or an IP
- **crtsh**: enumerates subdomains using [https://crt.sh/](https://crt.sh/)
- **port_scanner**: scans ports and identifies services (e.g. HTTP or FTP) on these ports
- **directory_index**: checks whether directory index is enabled
- **webapp_identifier**: checks what web application is running on a given port
- **sqlmap**: if an application is not a known CMS, checks for SQL injections
- **wp_scanner**: Checks e.g. whether a website is running an up-to-date version of WordPress
Alternatives (1/2)

**Osmedeus**

The data flow uses text files → hard to have a robust data flow.

**reNgine**

The purpose of the system is different, we would need to manage a fork.

**reconFTW**

Written in Bash, therefore hard to extend in a robust way.
Alternatives (2/2)

- **Nuclei** / Jaeles
  - Do not do reconnaissance.
  - Can be used as part of the pipeline if we already know the subdomains.

- **Naabu** / fingerprintx
  - Provides only a subset of information: ports and services on these ports.

- **nmap**
  - Even with script support it would be hard to adapt to e.g. enumerate subdomains.

- **zmap** / masscan
  - Solves one problem well: port scanning.
  - Not able to build pipelines out of multiple types of tools.

Used in the Artemis pipeline.
Scanning

We’ve been scanning the websites since January.

We have already scanned ~31k domains and IP addresses and ~85k subdomains.
Reported issues in 2023 so far (January-May)

~21.5k   SSL/TLS misconfigurations
~14.1k   SPF/DMARC misconfigurations
~9.2k    obsolete Joomla, WordPress or WordPress plugin versions
~5.4k    information leaks: AXFR, directory listing, phpinfo(), etc.
~1.5k    high/critical vulnerabilities from Nuclei or sqlmap

697      exposed backups, source code, database dumps or logs
75       exposed RDPs

~52.4k in total
Reporting and reactions

Our current workflow:

1. a package of reports is prepared semi-automatically
2. 1st line sends e-mails to best-known contacts
3. 1st line manages the follow-up communication (when needed)

Responses are mostly positive, but:

- they sometimes include bug reports (which are frequently correct!)
- sometimes the institutions report false positives
- sometimes we need to fix the contacts
- sometimes we are ignored
- sometimes the institutions fix the vulnerabilities without responding
Challenges

- Distinguishing true from false positives
  Example: if we detect that /wp-config.php.bak is present, we need to check whether it is indeed an exposed configuration file. We have lots of heuristics to keep the number of false positives low.

- Rate limiting in distributed environment
  Making sure no server is overloaded with requests is tricky with multiple modules.

- Scanning is slow
  The biggest cause is the per-host limiting behavior.

- Deduplication
  We need heuristics to detect whether two similar vulnerabilities on institution.com and www.institution.com are in fact one.

- Contact database
  Maintaining an up-to-date contact database requires significant effort.

- Running a non-trivial production service
  We have a medium-scale service where we sometimes need to troubleshoot unexpected administrative problems.

- Prioritizing the scans
Conclusion for administrators

Yes, they seem obvious - but following them would greatly decrease the number of problems found by Artemis.
Conclusion for administrators: updates

Detecting obsolete software versions with known bugs is easy.

Exploits for known vulnerabilities exist.
Conclusion for administrators: archived websites

Outside check allows to find archived or forgotten websites that can:

- use obsolete software (containing known vulnerabilities),
- be built without following of modern software engineering practices:

```php
query("SELECT * FROM posts WHERE id = " . $_GET["id"]);
```

Control what is exposed.
Conclusion for administrators: security by obscurity

Scanners can (and will) find:

- `/backup.zip` placed temporarily on the server,
- a test subdomain.
Conclusion for administrators: configuration files, logs, backups, code repositories...

https://[domain]:443/.git
https://[domain]:443/uploads/
https://[domain]:443/config.inc
https://[domain]:443/config.php.save
https://[domain]:443/configuration.php.bak
https://[domain]:443/configuration.php.save
https://[domain]:443/wp-config.php~
https://[domain]:443/wp-config.php.bak
https://[domain]:443/wp-config.php.old
https://[domain]:443/wp-config.php.save

...
Conclusion for developers: Roundcube misconfiguration - a case study
Conclusion for developers: Roundcube misconfiguration

Index of `/webmail/temp`

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td></td>
<td></td>
<td>Parent Directory</td>
</tr>
<tr>
<td>0a0</td>
<td>2020-02-17 14:13</td>
<td>6.8K</td>
<td></td>
</tr>
<tr>
<td>0a5</td>
<td>2019-12-05 22:52</td>
<td>8.2K</td>
<td></td>
</tr>
<tr>
<td>0be</td>
<td>2020-04-01 21:13</td>
<td>8.4K</td>
<td></td>
</tr>
<tr>
<td>0bc</td>
<td>2020-02-14 08:56</td>
<td>6.8K</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
<td></td>
<td>.thumb</td>
</tr>
<tr>
<td>0c1</td>
<td>2020-04-01 20:58</td>
<td>6.5K</td>
<td></td>
</tr>
<tr>
<td>0c3</td>
<td>2019-11-15 00:10</td>
<td>7.2K</td>
<td></td>
</tr>
<tr>
<td>0dc</td>
<td>2020-04-01 20:46</td>
<td>9.4K</td>
<td></td>
</tr>
<tr>
<td>0e6</td>
<td>2020-04-01 20:46</td>
<td>13K</td>
<td></td>
</tr>
<tr>
<td>0ea</td>
<td>2020-04-01 21:13</td>
<td>9.7K</td>
<td></td>
</tr>
<tr>
<td>0ed</td>
<td>2019-10-13 23:19</td>
<td>8.6K</td>
<td></td>
</tr>
<tr>
<td>0ed</td>
<td>2020-09-15 10:11</td>
<td>9.4K</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion for developers: Roundcube misconfiguration

Why is this not a good approach? What conclusions can we draw?
Conclusion for CSIRTs

- Unfortunately, there are still low-hanging vulnerabilities

- Many good offensive tools are available
  even plain Nuclei or WordPress/Joomla version check would find many vulnerabilities

- Not a huge project: currently 1 FTE: development + operations
  Managing contact list and sending prepared e-mails not included.

- 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.
It is easy to start a similar project and improve the security of your constituency.
Plans

- Develop the system:
  - add modules to detect new vulnerabilities
  - autoreporter - open source and make fully automatic
  - Regularly scan multiple groups of domains (including most popular .pl domains)
Slajd tytułowy bez zdjęcia. Można użyć również jako slajd kończący z podziękowaniami.

Questions?

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