CSIRT Tooling: Best Practices in Developing, Maintaining and Distributing Open Source Tools

CIRCL Computer Incident Response Center Luxembourg

2018-08-14

Contents

CSIRT Tooling: Best Practices in Developing, Maintaining and Distributing Open Source Tools

Introduction

The role of a CSIRT (Computer Security Incident Response Team) is key in information society and especially to improve cyber security in their constituencies and beyond. To achieve this, CSIRTs often have significant operational constraints such as limited budgets, a bound number of resources or/and overloaded staff members. Tooling, such as software or tools used in day-to-day activities in Digital Forensic, Incident Response and Threat Intelligence, offers CSIRTs to operate more efficiently the processing of constant flow of information and act in a timely manner (such as notifying victims, reporting, information sharing at European or International level or investigating technical evidences). In this document, a set of best practices is described to help CSIRTs to develop, maintain and distribute existing or new Open Source tools.

Definition

The term *CSIRT tooling* will be used in this document to describe any Open Source or Open Hardware project which aims to support digital forensic or incident response, specifically in the scope of the Computer Security Incident Response Teams.

Software Development Practices

A critical element in Open Source software development is the availability of the source code in a public repository along with the history of changes. Source control management such as *git* allows everyone to see the development activities (such as bug fixing and updates), to grasp the inner development practices of a tool and the community behind. While developing CSIRT tooling, a publicly[1] accessible source control management must be used.

A source control management repository for CSIRT tooling allows:

- To track the evolution and overall activity of a project including releases, versioning and patches
- To review the source code and ease the process of security assessment
- To monitor contributions and support new external contributions easily

Another significant benefit of relying on a public source control management such as *git* is to keep track of all changes in the commit messages associated with related changes. This allows to understand the background of a source code change and improve the overall level of documentation of a project. Such level of

[1] Git is just the core free software to handle the SCM (Source Code Management) repository. A CSIRT can operate a publicly-owned collaborative platform such as gitlab, gitea or rely on an existing public service such as GitHub.

transparency is an important requirement for tools and software which will be used for critical security operations. An issue tracker is also a must in order to allow users to easily report back bugs, open new features requests or ask specific question about the tool/software. It gives also insight if security vulnerabilities and assessment processes are properly implemented.

Software development methodologies also play a key role to attract new developers and contributors (from the CSIRT community or outside). CSIRTs developing tools or software should select the best methodology, fitting the project and/or their team's approach. There are formal methodologies for software engineering and more relaxed ones with an emphasis on the collaboration such as the PMF[2] model.

Recommendations

- CSIRT tooling must have at least one source control management repository (if the project is larger, multiple repositories might be required)
- CSIRT tooling repository must be publicly accessible
- CSIRT tooling repository must allow external contributors to propose changes (via pull-request) and open issues easily

Security Vulnerabilities and Software Assessment

CSIRTs play an important role in awareness raising for security vulnerability notifications. It's an opportunity for the CSIRTs to demonstrate the importance of security disclosure processes when producing software on their own. Having a specific section in the software documentation or main page describing the point-of-contact to report potential security vulnerabilities (with an associated PGP key) is a must.

If a vulnerability is found in the CSIRT tooling, a CVE assignment and publication are required to easily reach out your user base with precise information. Additional notifications to your users via different channels are often recommended (such as mailing-lists, social media or private notification to your constituency).

Regular security assessment (from manual review, automatic review or even fuzzing) of the CSIRT tooling is recommended to ensure a better overview of the current security state of the developed software. Documenting potential weak points with security countermeasures is a way to ensure that users operate the software in adequate conditions.

Recommendations

- CSIRT tooling must have at least a specific point-of-contact for security vulnerability notification with an associated PGP key

[2] Programming Methodology Framework aka PMF

- CSIRT tooling must provide a way to assign CVE in case of discovered vulnerabilities and provide a fix in a timely scope
- CISRT tooling must have a vulnerability disclosure policy

Open Source Software License

Choosing an open source license when publishing a CSIRT tooling is vital step and might impact the software on the long-term. A review of the CSIRT tooling might be required to ensure license compatibility:

- Is the CSIRT tooling relying on additional libraries or requirements?
- Are these requirements or libraries open source and compatible[3] with the foreseen open source license?
- If not, is there open source alternatives and are those compatible with the CSIRT tooling?

When selecting an open source license for your CSIRT tooling, multiple parameters might be involved such as:

- Do you need to ensure compatibility with other open source projects and especially CSIRT tools?
- Will your CSIRT tooling used a standalone software, will it be a library?

Recommendations

- CSIRT tooling must be licensed under an approve open source license[4] [5]
- CSIRT tooling must regularly review the license compatibilities with dependencies

Privacy and Personal Data Processing

The aim of developing CSIRT tooling is to support Incident Response teams in their day-to-day activities. Processing incident-related or threat intelligence data might include personal information. It is strongly recommended to review the privacy implication of developing and also distributing such a tool. In order to clarify the scope and help potential users to ensure lawfulness, a review of the privacy implication might be included along with the CSIRT tooling (see AIL software privacy review[6] as an example or the one for MISP called Information sharing and cooperation enabled by GDPR[7]).

[3] FSF list of license compatible with the GNU GPL
[4] FSF list of license compatible with the GNU GPL
[5] Approved licenses from the open source initiative
[6] AIL information leaks analysis and the GDPR in the context of collection, analysis and sharing information leaks
[7] Information sharing and cooperation enabled by GDPR

Recommendations

- CSIRT tooling should include a description of the information which has a privacy impact and how to improve privacy when deployed.
- CSIRT tooling should include functionalities and technical measures in order to improve privacy.

Contribution and Collaboration

Attracting contributors is a key element for a CSIRT tooling to assure sustainability of the project. Contributors often start as a user of the CSIRT tooling. A welcoming environment fosters users to contribute and collaborate more in a project. In order to reach such a collaborative environment, a code-of-conduct might help to attract more contributors and collaboration within a CSIRT tooling project.

Recommendations

- CSIRT tooling should include a code-of-conduct such as *Contributor Covenant Code of Conduct*[8].

Interoperability

Another significant adoption of CSIRT tooling is the ability to integrate the tool easily in the tool-chain used by CSIRTs. Interoperability can be expressed in various ways:

- A documented ReST API
- Reusing existing free and open standards
- Reusing existing data models or data representation (such as the common taxonomies[9] or misp-objects[10])
- Integrated connectors with existing CSIRT tooling

The sustainability of a CSIRT tooling often depends on the ability of users to integrate the software with legacy systems or existing processes. The more options available to the users will ensure a constant usage of the CSIRT tooling. If the CSIRT tooling reaches a significant maturity, publication of the format can be submitted to standardisation process such as IETF, ITU or OASIS (such as the MISP-rfc format[11]).

[8] A Code of Conduct for Open Source Projects
[9] MISP Taxonomies is a set of common classification libraries to tag, classify and organise information
[10] MISP object template definition generated from specification files
[11] The specification and formats used in the MISP project

Recommendations

- CSIRT tooling should have an open and documented API to interact with tools (such import/exporting information, triggering operations of the CSIRT tooling)
- CSIRT tooling may publish their format specifications to a standard organisation such as IETF, ITU or OASIS.

Conclusion

Developing CSIRT tooling within an open source methodology allows to reach new audiences, to bring new use-cases and to ensure improved integrations with existing practices in CSIRT. The initial work to release as an open source can introduce some additional work but the benefit is often larger than the initial cost. The objective of this document is to list all the best practices in the field.

References

Existing CSIRT tooling

Software	CSIRT lead	Location
MISP	CIRCL	https://www.misp-project.org/
AIL	CIRCL	https://github.com/CIRCL/AIL-framework
BGP Ranking	CIRCL	https://github.com/D4-project/BGP-Ranking
cve-search	CIRCL	https://github.com/cve-search/
IntelMQ	CERT.at	https://github.com/certtools/intelmq
n6	CERT.pl	https://github.com/CERT-Polska/n6
TheHive	BDF CERT	https://github.com/TheHive-Project/TheHive
Cortex	BDF CERT	https://github.com/TheHive-Project/Cortex-Analyzers/

Revision of the document

- Version 1 - Team CIRCL

Acknowledgment

The contributors to the document are:

- Alexandre Dulaunoy - CIRCL
- Sascha Rommelfangen - CIRCL
- Gerard Wagener - CIRCL

This work was partially funded by CEF (Connecting Europe Facility) funding under CEF-TC-2016-3 - Cyber Security *Improving MISP as building blocks for next-generation information sharing*.

Co-financed by the European Union
Connecting Europe Facility

Contact and Collaboration

If you have any question or suggestion about this topic, feel free to contact us. This document is a collaborative effort where external contributors can propose changes and improvement to the document.