SCRATCh

Guidelines and Best Practices

Deliverable

Deliverable 4.3: Best Practices & Guidelines

Work package: WP4

Affected milestone: MS6

Partners involved: Almende

Date: 7/3/2022

Deliverable version: v1.0

Editor: Merijn van Tooren, Almende

Author(s): Merijn van Tooren, Almende

Responsible Contact: Merijn van Tooren

Almende

Stationsplein 45 Unit D1.116, 3013 AK Rotterdam, NL

merijn@almende.org

+316-10657644

Version history

Date	Version	Author	Comment
31/3/2020	v0.1	Merijn van Tooren	Structural Setup & Introduction
22/4/2020	v0.2	Merijn van Tooren	Add examples
14/10/2020	v0.3	Merijn van Tooren	Add analysis of existing material
8/9/2021	v0.4	Merijn van Tooren	Chapter 3 on challenges and generic guidelines
3/3/2022	V1.0	Merijn van Tooren	Completion of document

Table of Contents

1. Introduction

1.1. Terminology

Within this deliverable, there is discussion of the terms “guidelines” and “best practices”. In order to facilitate communications, these terms will be briefly distinguished here:

- A guideline is defined as a large-scope, abstract element of advice. For example: *“Prioritise completeness over depth. Use standards and checklists to ensure that no security aspect is left untreated.”*
- A best practice is defined as a small-scope, specific set of actions. For example: *“Move any unprotected public network traffic into a VPN tunnel for basic protection.”*

1.2. Introduction Proper

The objective of this deliverable is to document the guidelines and best practices for secure DevOps for IoT as invented within the SCRATCh project. During this research track, there is an examination of many existing standards, guidelines and tools, as well as of the DevOps process as a whole, the roles security plays in that, and development of new tools. This yields new insights into best practices. Furthermore, SCRATCh investigates the specific case of introducing security in DevOps “from scratch”, in SMEs that as of yet have not integrated security in their development process. This more specific case of secure development warrants guidelines that are user friendly and optimize security with limited complexity and effort.

2. Existing Best Practices & Guidelines

In order to make a complete study of best practices and guidelines for SCRATCh and related use cases, it is prudent to start with an investigation of the existing material. To this end, the SCRATCh project has created an online database that combines the most prominent current standards for perusal. This can be found at https://scratch-itea3.eu/ following the SOTA drop-down menu found in the top right, leading to e.g. https://scratch-itea3.eu/sota/good_practices/list.

Notably, from ENISA, there is a set of “Good Practices”, of which the url has been provided above. Each of these entries brings together a number of guidelines for cyber security. Generally, these are applicable to SCRATCh no less than to other security themes, and they constitute good examples of what SCRATCh considers a guideline.

They are not necessarily universal: for example, the guideline “Restore Secure State - Enable a system to return to a state that is known to be secure, after a security breach occurs or if an upgrade is not successful.” may not be possible to follow in every situation. An attacker may gain full control of the system and prevent such a rollback from happening, even if it has been implemented, or the ‘secure state’ a system returns to may be compromised persistently e.g. by the attacker obtaining keys or a physical manner of tampering with the device.

Thankfully, the ENISA Good Practices are also mostly guidelines that are very understandable even with little security expertise, which makes it feasible to recommend reading these guidelines to SMEs when setting up and performing SecDevOps. Many of the guidelines can also be followed up on with little specialised security expertise. There are more sets of requirements from ENISA on the SCRATCh online database, and they have similar yield, but are less nicely organised for overview.

It can also be recommended to familiarise oneself with the list of Threats that has been provided by ENISA. As of the time of writing, a general sense exists within the SCRATCh project that awareness of threats is the first and foremost concern when aiming to secure a system, and particularly SMEs without security experts may all too easily overlook some of these threats without a comprehensive list.

There are also standards from ETSI, which constitute a much more narrow and shallow set of requirements for overall SecDevOps, but include important considerations about managing accounts and data for customers. These may not be equally relevant to every IoT-heavy use case, but are worth looking into when user accounts and user data are involved.

From OWASP, very helpfully, there is a Top 10 of security gaps found in studies of existing systems. By focusing on vulnerabilities that are easy to understand and address, and keeping the list short, OWASP has hereby created a very powerful guide for SMEs with limited security expertise. It would seem prudent for SCRATCh to recommend regular reviewing of this Top 10 to avoid these pitfalls. Also found in the SCRATCh

database is the OWASP ASVS, but this collection is very large and more suited to searching than use as an introduction or checklist.

More lists of requirements have been found and made available on the SCRATCh online database, but a SME systems designer will quickly start to encounter redundant requirements as well as ones that are specific and out of scope for them.

In conclusion, while all of the standards found are relevant, some of the sets of requirements may prove much more efficient and accessible to SMEs than others, especially during early stages and when performing security by design. At this stage, SCRATCh recommends thorough usage of the ENISA threat and good practice lists, and frequent usage of the OWASP Top 10 vulnerabilities list. Apart from these observations, no few requirements stood out as particularly relevant to SCRATCh over the others, and comprehensiveness is considered key rather than specific security aspects.

3. SecDevOps from SCRATCh

3.1. Challenges

During this project, research has been done into the reality experienced by various small and medium enterprises when they are confronted with DevOps and SecDevOps. In short: the outlook is rather bleak for these parties. From the outset, many of the small development teams in the modern world are unaware of any form of DevOps beyond a ubiquitous, simplified form of Scrum practices based around weekly goals and repetitive update meetings. Integration of tests is hardly commonplace, let alone unit tests, automation or an audit process.

Cybersecurity is more of a dream than a pursuit, at this stage. Enterprises readily acknowledge that cybersecurity is important, even essential, and particularly for their own industry too, yet they leave this matter largely unattended. Most small teams do not have a security expert, which is unsurprising, as it would be quite a costly overhead for a small team; there are not as many security experts on the market as there are small development teams, either. This, however, is also the root issue: without a security expert, a team is ill-equipped to understand the many threats and following security requirements that should be managed. Without an understanding of cybersecurity, there can be no creation of SecDevOps.

As it seems unlikely that there will ever be a security expert for each development team in business, solutions must be found in other directions. As demonstrated before, there are helpful standards and guidelines available, but it is not trivial to select the right guidelines and be able to fully comprehend them. The matter of knowing which guidelines and regulations to focus on, and the ability to understand them quickly, are essential points upon which the industry can improve. A similar case can be made for SecDevOps as such: one or more popular and accessible process standards may be able to improve the overall practices of such enterprises.

A special mention must be made of security requirements. As it stands, methods of finding, documenting and monitoring security requirements are limited. Such solutions either demand a great overhead from the team, or fail to properly oversee the nuances of such requirements. Better management of security requirements would greatly improve all SecDevOps, even for large enterprises.

3.2. Generic Guidelines

These are some basic understandings of SecDevOps that have held since early stages of the project:

- <Pick the low-hanging fruit.> It is impossible to be 100% secure, and it is impossible to have a perfect process, and it is impossible to take every possible measure within a realistic budget. Therefore, it is imperative to select solutions based on their costs as well as their results, and one should start by covering a breadth of topics with simple fixes.

- <Conceal your vulnerabilities.> Attackers will take the cheapest avenue of offense that they can, and prefer to strike when they are certain of an easy victory. Even the simplest forms of communication security and code obfuscation can greatly impede a hacker's understanding of your system, slowing them down or completely dissuading them.
- <Document your security requirements.> Even if it is difficult and prone to error, it is vital to document and maintain the security requirements for a project. It is all too easy to lose several along the way, or reinterpret requirement as time goes on, unintentionally leaving holes in the defence of the resulting product.

4. Best Practices

This section sequentially details the Best Practices postulated in SCRATCh. Each best practice is named, assigned a relevancy to part of or the entire DevOps process, and explained.

- Check CySec Standards (Plan, possibly later): Filter ENISA and OWASP lists for security requirements relevant for your project. Err on the side of too much rather than too little. This will yield a rudimentary list of concerns that should all be considered, if not addressed. Do this as early as possible, but it is never too late.
- Build Chain (Plan, Build, Test, Release, Deploy): Set up an automated build chain, typically using GitHub and/or Docker. Use this time early on to design a build chain to integrate with. This will make testing and releasing go much more smoothly in the long run, and prevent later speed bumps.
- Integrate Secure Hardware Elements (Plan, Deploy): If feasible, integrate a secure hardware element in edge and end devices in order to protect your devices from undesired updates and queries.
- Code Obfuscation (Code): Between writing and building, change code so that it retains its function but is much harder to understand and reverse-engineer.
- Prioritise Tests (Test): Track the expended time and energy, as well as the rate of outcomes for each test, and re-prioritise tests so that issues are more likely to be detected early than late in the phase.
- Update Manifests (Deploy): Attach metadata to update deployments in order to ascertain that updates are only installed on the right hardware, within a suitable timeframe from deployment, and on a system that will be improved by it.

5. Specific Guidelines

This section details SCRATCh guidelines that are specific to certain parts of the DevOps process. They are sorted into subsections to organise by way of this distinction. This is done according to the SCRATCh “3C Method” which splits the DevOps cycle into Constrain, Comply and Control consecutively.

Constrain

- Manage Security Requirements: Put considerable effort into creating a good overview of all security requirements, and prepare for the execution of the project. The requirements list must be easy to access, search and edit, and the requirements should tie into the Comply phase heavily.
- Security By Design: Where possible, determine security measures before writing code, or even before choosing hardware. Being able to adapt design to security measures can make the security measures much cheaper and easier to implement, and even more so to maintain.

Comply

- Automate Tests: Create test suites that run themselves. Create tests that automatically run on build, or on upload. Where possible, automatically test based on listed security requirements. The greatest risk is to simply forget about a certain threat.
- Secure Deployment: Prevent others from blocking your software deployments, and especially from abusing your software deployment channels. Engineer a deployment method that is sustainable and automatic. Regular updates are the lifeblood of cybersecurity.
- Manage Keys & Certificates: Take care that keys and certificates used to sign software are not hijacked. Certificate chains can help to great degrees by introducing layers of clearance and the ability to revoke.
- Analyze Traffic: Log network traffic in and around your solution and check for vulnerabilities, e.g. exposed passwords.
- Vulnerability Scans: Scan for common known vulnerabilities in code and imported libraries. Look into “Common Vulnerabilities and Exploits / CVEs” and “Software Bill of Materials / SBOM”.

Control

- Deceive Attackers: Use honey traps to catch attackers and proxies to intercept typical attacks. Most attacks are cheap and fall for the simplest tricks, and this will greatly increase your effective awareness.
- Monitor for Anomalies: Keep track of activity, power usage, network traffic, etc to detect anomalies that may point to unwanted interference.

6. Summary

The SCRATCh project has allowed for a dive into the world of SecDevOps, enough to get an idea of its vastness and depth and gain an understanding of many of its facets. As is befitting of research, it leaves more questions than it ever answered. During the project, a new SecDevOps methodology has been defined that will hopefully prove more accessible and practical in nature than others before it.

Many best practices and guidelines have been identified and reiterated upon in this document. It is considered vital to carefully establish, manage and maintain security requirements throughout a project so as not to leave any threats unaddressed. It is the weakest link that will break the chain, and in cybersecurity, the easiest attack will be the one suffered.

In closing, it is concluded that there are many ways to optimise SecDevOps, and ultimately choices must be made based on the nature of the specific project being improved. A method to help choose measures to implement would be a desirable follow-up to these findings.