IT-Grundschutz and DevSecOps

An analysis of the requirements for DevOps in the German BSI IT-Grundschutz

Moritz Lübken
Erasmus Exchange Student
KTH Stockholm, Sweden
moritz.luebken@rwth-aachen.de

***Abstract*—The BSI IT-Grundschutz is the go-to standard for German authorities when it comes to cybersecurity. It is compliant with the widely used ISO/IEC 27001 norm and includes specific requirements for a broad range of domains. This essay analyzes the requirements concerned with software development and DevOps, compares them to other frameworks and standards and proposes some additions.**

***Index Terms*—IT Grundschutz, DevSecOps, Cybersecurity**

I. Introduction

In the light of the slow but steady digitization of the German government and its interaction with the citizens there is a growing number of applications developed by or for the government or its agencies. When it comes to cybersecurity the IT-Grundschutz standard [1] that has been published yearly since 1994 by the Federal Office for Information Security (BSI) is the relevant norm for all governmental agencies. It is compliant with the ISO/IEC 27001 standard and can be certified by the BSI and its independent auditors. Compared to the ISO/IEC 27001 the IT-Grundschutz states very specific requirements for a wide range of domains and, while the main focus of the standard lies in organizational aspects and in building an information security management system, it contains some deeply technical requirements too. The requirements are structured into modules and classified as either basic, standard or for increased protection needs. While the current version is only available in German, the 2021 version can be accessed in an English version [2].

> The following HTTP headers SHOULD be used at minimum: Content-Security-Policy, Strict-Transport-Security, Content-Type, X-Content-TypeOptions, and Cache-Control.

Figure 1. CON.10.A14 Secure HTTP Configuration of Web Applications: A very technical requirement

Looking at the facts that software projects become increasingly important for authorities and that the IT-Grundschutz standard already contains very specific requirements for software development, the question arises, whether these requirements are sufficient and are up-to-date regarding the latest techniques in DevOps and software development.

II. Methodology

To find potential weaknesses of the IT-Grundschutz alternative standards and frameworks are identified. Due to the limited scope of this essay this is done by a simple web search. Looking at all requirements the standards would exceed the scope of this essay so this work focuses on requirements with a technical component. This means that requirements for purely organizational or conceptual aspects are disregarded. The IT-Grundschutz contains concrete technical requirements and thus should be comprehensive and up-to-date.

For the identified requirements in other standards a text-search is conducted in the requirements of the IT-Grundschutz. If the text-search finds results in the document it is manually verified, that the requirements are the same or that the IT-Grundschutz requirements are at least as comprehensive.

For all requirements where the text-search cannot find any results or where the manual verification does not find the requirements to be the same an additional manual matching was tried within the relevant modules.

III. Identifying relevant standards

The following standards, frameworks or guidelines with a connection to DevSecOps could be identified by a search on the web. As explained in the previous section this list is in no way complete and when it comes to software security there is a whole number of other security standards with no direct connection to DevOps.

A. OWASP DevSecOps Verification Standard

The standard lists 40 requirements of which 32 are considered as technical enough for the purpose of this essay. [3]

B. Microsoft Secure Development Lifecycle

Microsoft offers an Excel-Spreadsheet with 22 mostly organizational higher-level requirements. However the list also contains some concrete steps like ”Perform Fuzz Testing”. [4]

C. OWASP Devsecops Maturity Model

The standard lists over 160 different requirements for different levels in the categories ”Build and Deployment”, ”Culture and Organization”, ”Implementation”, ”Information Gathering” and ”Test and Verification”. [5]

Since the list is quite comprehensive and in parts repetitive due to the different maturity levels it was just manually scanned for additional requirements that were not yet included in the requirements of the other two sources and where a direct mapping to the IT-Grundschutz requirements was not obvious.

IV. Comparison with the IT-Grundschutz

For a list of 47 specific, technical, and non-overlapping requirements a mapping to the IT-Grundschutz compendium 2023 was attempted. For eleven of them a full match could be found and for another 11 at least partial matches were found[1]. The full matches include requirements to the documentation of software, the security of the development environment, the usage of static code analysis, some cryptographic requirements and points concerning vulnerabilities in dependencies[2].
The detailed results of the partially matched requirements can be found in the attached Excel spreadsheet. Most of them do not call for major changes but rather for small adjustments or explanations. The following two however seem worth mentioning:

A. Hardcoded Secrets Detection

CON.8.A5 states that all information that is not relevant for production like comments with secrets should be removed before deployment. However it does not include that the detection of these secrets should be automated. Using scanners for hard-coded secret detection is relatively easy and there are free open-source tools available. The likelihood of exploit is very high [6] and the weakness is very common, as demonstrated by Meli et al. who found hard-coded secrets in 0.005% of all files they scanned on Github. That seems like more than enough reason to add a requirement for an automated hard-coded secret detection.

B. Application Security Logging

While OPS.1.1.5.A6 requires a centralized logging infrastructure this does not seem connected to software at all. The software specific requirement CON.8.A5 states that events that are relevant to the security of the application have to be logged but it does not include anything on a centralized infrastructure or real-time monitoring which is required in the DevSecOps Verification Standard [3].

For the requirements without a match a full list of proposals for the IT-Grundschutz can be found in Appendix A. The following section will discuss the most important ones and give some justification for why the requirements should be included in the standard.

C. Software License Compliance

In APP.6.A9 a inventory of used software and their licenses is required. However the module is not connected to software development but to software used in organizations in general. While even if one would apply this to dependencies in a software project it does not say anything about automation.

[1]A partial match here means, that there is a requirement in the IT-Grundschutz compendium that covers the topic but misses some important parts.

[2]The term SCA is never used and the requirement CON.8.A20 does not say anything about automation but it says that all dependencies have to be scanned for vulnerabilities or outdated versions.

D. Inline IDE Secure Code Analysis

No requirement could be matched to the Inline IDE secure code analysis stream of the OWASP DevSecOps Verification Standard. There are tools available for most modern programming languages that can run static code analysis during the coding process and show warnings or errors in the IDE. These ”linters” often can be configured to not just enforce security policies but even coding style. Some of the reasons to use such tools have been collected in [7].

E. Dynamic Application Security Testing

While CON.8.A7 mentions static analysis nothing about dynamic application security testing can be found. The requirements for testing seem to put the focus on manual testing and although OPS.1.1.6.A5 states that security-related tests have to be performed it does not specify this and responsible for this requirement is not the developer of the software but the responsible person at the organization using the software. Static analysis can help prevent a lot of vulnerabilities but some simply cannot be found in a static setting. That is why dynamic application security testing should be performed and there are solutions out there. One example implementation is described in [8]. The tests should also include fuzzing techniques to increase the chance of finding certain errors. Li et al. even state that ”Fuzzing is currently the most effective and efficient vulnerability discovery solution.” [9, p. 12]

F. Automated Dependency Updates

CON.8.A8 includes that dependencies have to be updated and patches provided, if any dependencies publish security updates. However it does not say that this has to be done in an automated way and the requirement is quite unspecific. ”Vulnerable and outdated components” is ranked number six of the OWASP Top 10 [10] with a maximum incidence rate of 27.96%. That means that one of the corresponding CWEs could be mapped to more than every fourth application scanned. In a 2018 study researchers looked at 269 reported vulnerabilities in NPM packages and found 72,470 vulnerable packages depending on them [11].

V. Conclusion

It is clear that the IT-Grundschutz standard is not up-to-date when it comes to software development and DevOps. By a simple comparison with only three other frameworks 21 requirements could be identified that are completely missing in the compendium. For many of them there seems to be evidence that they significantly improve security.
The word „pipeline“ can be found exactly three times in the latest compendium, “CI/CD” nine times. All findings are in the modules “Kubernetes” and “Containerization” and the only requirements concern the security of containers used for CI/CD pipelines.
The BSI should definitely work on improving the standard and keeping it up to date. However it seems very hard to maintain a standard that covers all areas of information technology. The NIST e.g. offers the Secure Software Development Framework

(SSDF) next to the NIST Cybersecurity Framework and is currently looking into DevSecOps to develop a new framework [12]. This might be the more reasonable approach but if the BSI aims to keep it one standard, looking at the amount of software projects that have to be realised for a digitized government, it is necessary to improve the software and DevOps security requirements.

I certify that generative AI, incl. ChatGPT, has not been used to write this essay. Using generative AI without permission is considered academic misconduct.

References

[1] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz Kompendium*. Feb. 2023. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Kompendium/IT_Grundschutz_Kompendium_Edition2023.pdf?__blob=publicationFile&v=4#download=1.

[2] Federal Office for Information Security. *IT-Grundschutz-Compendium*. Feb. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Grundschutz/International/bsi_it_gs_comp_2021.pdf?__blob=publicationFile&v=4.

[3] Jamieson O'Reilly and Yudhi Yudhistira. *OWASP DevSecOps Verification Standard*. 2023. URL: https://owasp.org/www-project-devsecops-verification-standard/.

[4] Microsoft. *Microsoft Secure Development Lifecycle*. URL: https://www.microsoft.com/en-us/securityengineering/sdl.

[5] Timo Pagel and Aryan Prasad. *OWASP DevSecOps Maturity Model*. 2023. URL: https://owasp.org/www-project-devsecops-maturity-model/.

[6] Mitre. *CWE - CWE-798: Use of Hard-coded Credentials (4.11)*. https://cwe.mitre.org/data/definitions/798.html. (Accessed on 05/02/2023).

[7] Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie van Deursen. "Why and how JavaScript developers use linters". In: *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2017, pp. 578–589. DOI: 10.1109/ASE.2017.8115668.

[8] Thorsten Rangnau et al. "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines". In: *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*. 2020, pp. 145–154. DOI: 10.1109/EDOC49727.2020.00026.

[9] Jun Li, Bodong Zhao, and Chao Zhang. "Fuzzing: a survey". In: *Cybersecurity* 1.1 (June 2018), p. 6. ISSN: 2523-3246. DOI: 10.1186/s42400-018-0002-y. URL: https://doi.org/10.1186/s42400-018-0002-y.

[10] OWASP. *A06 Vulnerable and Outdated Components - OWASP Top 10 2021*. https://owasp.org/Top10/A06_2021 - Vulnerable_and_Outdated_Components/. (Accessed on 05/04/2023).

[11] Alexandre Decan, Tom Mens, and Eleni Constantinou. *On the Impact of Security Vulnerabilities in the npm Package Dependency Network*. https://dl.acm.org/doi/pdf/10.1145/3196398.3196401. (Accessed on 05/04/2023). Software Engineering Lab, University of Mons, 2018.

[12] NIST. *DevSecOps — CSRC*. https://csrc.nist.gov/Projects/devsecops. (Accessed on 05/04/2023).

Appendix A: Proposed Requirements

Security Reporting
Security findings from multiple sources SHOULD be collated into a single document and populated into a real-time dashboard.

Security Policy and Regulatory Compliance
Compliance SHOULD be verified in real-time, and the findings SHOULD be tracked in a centralized issue tracking system. The compliance status SHOULD be enforced and periodically reviewed.

Security Requirements and Standards
Compliance with industry security standards and best-practices SHOULD be verified in real-time. They SHOULD be enforced and periodically reviewed.

Manual Secure Code Review
Coding standards SHOULD include a security checklist and security requirements. They SHOULD be used for peer-reviews. The coding standards SHOULD be reviewed periodically.

Software License Compliance
When using third-party libraries, a tool to perform scans for license violations SHOULD be included in the CI pipeline. The findings SHOULD be automatically tracked in a centralized issue tracker and the tool's effectiveness SHOULD be reviewed periodically.

Inline IDE Secure Code Analysis
Plugins for inline code-analysis / linting SHOULD be used. Rules for security, secret detection and code style SHOULD be centrally managed. Commits of insecure changes to the codebase SHOULD be prevented.

Container Security Scanning
A tool for container vulnerability analysis SHOULD be included in the build pipeline if containers are used. The findings SHOULD be tracked in a centralized issue tracker and its effectiveness SHOULD be reviewed regularly.

Dynamic Application Security Testing (DAST)
The pipeline SHOULD include a tool for dynamic application security testing and the results SHOULD be taken into account before a deployment. In addition, the findings SHOULD be tracked in a centralized issue tracker and the effectiveness SHOULD be regularly reviewed.

Security Test Coverage
Addition to OPS.1.1.6.A12: The test coverage SHOULD be reported to a centralized issue tracking system and continually increased.

Secret Management
If secrets have to be stored they SHOULD be stored in a centralized secrets store. This SHOULD be regularly reviewed and a rotation schedule for the secrets SHOULD be used.

Infrastructure-as-Code (IaC) Secure Deployment
If a project uses infrastructure configuration files they SHOULD be using version control and a release automation process SHOULD be in place. The least privilege principle SHOULD be applied to all deployments of infrastructure changes.

Compliance Scanning
Software projects SHOULD use tools that automatically run compliance scans. These SHOULD be reported to a centralized issue tracking system.

Perform Fuzz Testing
The Dynamic Application Security Testing should include "fuzz testing" techniques.

Archive all Release Data
Together with the release artifacts all specifications, source code, binaries, private symbols, threat models, documentation, emergency response plans, license and servicing terms for any third-party software and any other data necessary to perform post-release servicing tasks SHOULD be archived.

Signing of code
All commits into the repository SHOULD be signed.

Inventory of dependencies
The developers of a software project SHOULD keep an up-to-date inventory of their dependencies. This SHOULD contain the documentation for the decision process that this dependency is safe to use and the results from vulnerability scans.

Deployment without downtime
Software updates SHOULD be possible without any downtime by using techniques like rolling updates.

Automated dependency updates
Tools for automated dependency updates SHOULD be used. If this is technically impossible a concept SHOULD make sure that dependencies are kept up to date.

Maximum image lifetime
Containers SHOULD have a maximum lifetime before they are restarted with new images to decrease the risk of manipulated images.

Use of a chaos monkey
When the requirements for availability are high methods of Chaos Engineering SHOULD be used to ensure the resiliency of the system.

Security Unit Tests
Unit tests for security aspects MUST be run automatically before a release.