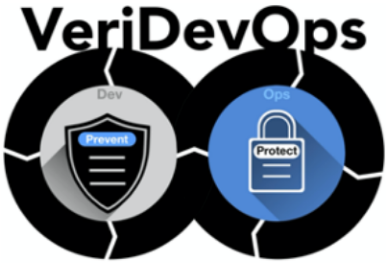
D2.4 Requirements patterns matching tool chain

Andrey Sadovykh, Etienne Brosse, Alessandra Bagnato



VeriDevOps

Automated Protection and Prevention to Meet Security

Requirements in DevOps Environments

D2.4 Requirements patterns matching tool chain

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957212. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Contract number:	957212
Project acronym:	VeriDevOps
Project title:	VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps Environments
DELIVERY DATE	30/09/2022
Author(s):	Andrey Sadovykh (SOFT), Etienne Brosse (SOFT), Alessandra Bagnato (SOFT)
Partners contributed:	SOFT
Date:	30/09/2022
Version	1.0
Revision:	1
Abstract:	This report accomplies the software deliverable 2.4 that includes the tool chain to map the requirements and patterns including the security recommendations.
Status:	Submitted

DOCUMENT REVISION LOG

VERSION	REVISION	DATE	DESCRIPTION	AUTHOR
	01	15/09/2022	Draft	SOFT
	02	22/09/2022	Revision after the internal review	SOFT, IKER
	03	29/09/2022	Revision after the internal review	SOFT, MDU

TABLE OF CONTENTS

Executive Abstract

VeriDevOps project [1] automates the analysis and verification of security requirements as well as hardens security of systems in operations while supporting the DevOps context with Continuous Integration and Delivery/Deployment (CICD) scenarios.

With this regard, the current deliverable outlines a toolchain that is dedicated to automation of security requirements analysis and identification of related recommendations, tests, and fixes through a specific mapping method. This approach addresses the requirements analysis stage of the DevSecOps process. There are very few methods and tools that deal with this stage. In VeriDevOps we provide practical solutions and automation methods that are illustrated examples.

The mapping mechanism relies on ARQAN and RQCODE tools by SOFTEAM. ARQAN applies machine learning (ML) for natural-language processing tasks. ARQAN may classify a text as a requirement and identify the relations to the security domain. In addition, ARQAN is capable of semantic search functionality that intends to "understand" a requirement and locate semantically-close recommendations from a repository. Currently, ARQAN is pre-trained for the Security Technology Implementation Guidelines (STIG) repository and IEC62443 set of recommendations.
RQCODE is a novel approach that applies Seamless Object-Oriented Requirements (SOOR) paradigm to security requirements. RQCODE proposes to implement SOOR in Java language. In particular, temporary requirements patterns and a large set of STIG guidelines are translated into Java with the SOOR paradigm in mind. That way, Object-Oriented Programming (OOP) analysis can be applied to the collections of security requirements. Moreover, the security requirement verification and validation may be integrated into Continuous Integration and Delivery (CI/CD) pipelines.

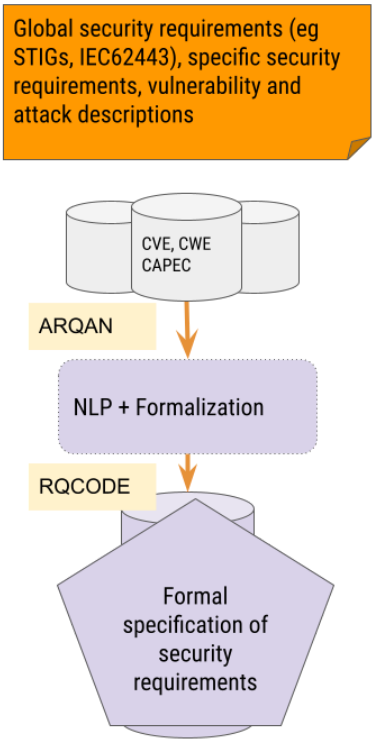
We demonstrate the mapping scenarios in the context of CI/CD pipelines whereas the requirements are entered by developers in the issue tracking systems and analysed by the automation bots. These bots label the requirements as related to security to highlight their importance. Moreover, the bots indicate semantically related STIGs and suggest related RQCODE to automate security testing.

The document outlines the work in progress for automation of the security requirements analysis. The evaluation by the end-users in WP5 will indicate areas for further improvement.

1. Introduction

In VeriDevOps, ***Work package 2 - Automated generation of security requirements*** investigates automatic extraction, formalisation and verification of the security requirements from natural language requirements, vulnerability databases and standards. The resulting information is used as input for WP4 - Prevention at development and WP3 - Reactive Protection at Operations.

The second task of WP2, ***T2.2: NLP for Requirements Formalisation*** explores the natural language processing methods for the automated generation of security requirements and their formalisation. The task explores open data sets (e.g. [2]) and case studies specifications to extract requirements, to provide recommendations for their classification and to match those requirements with formal specification patterns. The current deliverable, ***D2.4: Requirements patterns matching tool chain,*** presents the resulting toolchain to support NLP methods in Continuous Integration Continuous Delivery (CICD) pipelines. The toolchain is presented from the end-user perspective.



The current deliverable extends the concepts and tool results that were reported in the previous deliverables:

- D2.3 Requirements automated generation toolchain [3]
- D2.7 Patterns catalogue [4]
- D1.6 VeriDevOps Framework - initial version [5]

Note: The work on the matching mechanism is under active development and will be continued till M36 - end of the project. The current deliverable explains the conceptual principles of the matching mechanism, provides the current vision of the CICD pipelines involving NLP for security requirements analysis and recommendations, as well as the current state of the development of the supporting tools for the pipeline.

1.1. VeriDevOps GitHub Repository

GitHub is the go-to choice for open source projects with a community of 83+ million developers and 4+ million organisations including 90% of Fortune 100 companies. Therefore, we decided to create a central repository on this platform that is grouping all the tools, prototypes, and experiments. Some of those projects are readily available and public, some are not.
The current release of the VeriDevOps Framework can be available here [6]. As part of this framework, the following tools demonstrate the concepts and results reported in this deliverable:

- ARQAN [7] - demonstrator for NLP tools for the analysis of the security requirements
- RQCODE [8] - catalogue of security requirements patterns, tests, and fixes.
- ARQAN.services [7] - microservices for ARQAN NLP tools that are required to automate security requirements analysis in the CICD scenarios.
- CICD ARQAN RQCODE with GitHub Actions [9] - implementation and a demonstrator for the CICD scenarios to automate security requirements analysis. This implementation fits for the open source projects fully relying on the GitHub infrastructure.
- CICD ARQAN RQCODE with Jenkins [10] - implementation and a demonstrator for the CICD scenarios to automate security requirements analysis. This implementation is an enabler to integrate ARQAN and RQCODE in local or private environments, which is more suitable in an industry context.

Each repository includes the installation guidelines as well as a user's guide.

1.2. Feedback form

VeriDevOps users and the community are welcome to leave bug reports and feature requests in the Issue tracking system for each tool.

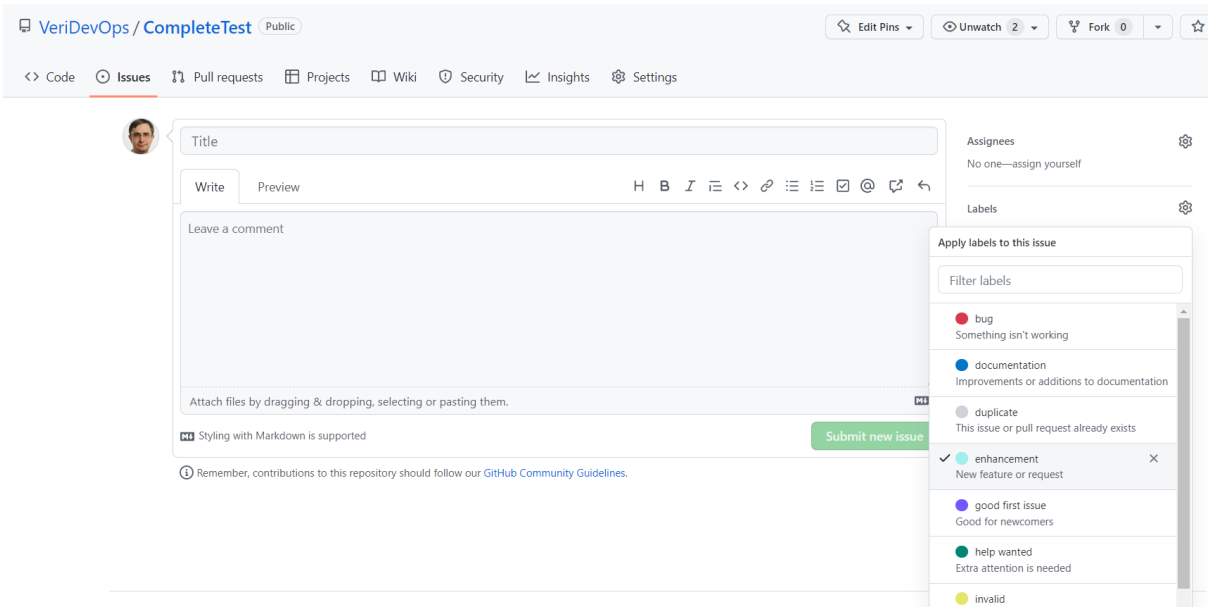


Figure 1. Feedback form for Issues tracking.

2. Bringing Security Requirements Analysis To The Early Stages Of A Project.

The current DevSecOps methods [11] integrate the security aspects into the DevOps context, i.e. cybersecurity has to be addressed at all the DevOps stages - plan, develop, deliver, monitor. DevOps promotes higher automation of all the quality-related activities - code reviews, static analysis, dynamic analysis, monitoring. Yet, this automation for the security aspects is mostly focused on the stages where the code is already available or even when the system is already in the production environment. Many authors [12], [13], [14] underline that dealing with non-functional requirements and, in particular, cyber-security at later stages may lead to higher costs and higher cybersecurity risks. Thus, it is important to introduce automation in the earlier stages - starting from requirements analysis.

Standardised requirements and recommendations play a critical role in developing secure and trustable systems. These standards address various systems and architectural layers:

- IEC 62443 standard deals with the Industrial Internet Of Things (IIOT) domain [15];
- Security Technology Implementation Guidelines (STIGS) provide recommendations for extensive list of platforms at the operating system and application level [16];
- Open Web Application Security Project (OWASP) provides best practices and guidelines for web and mobile applications [17].

Knowing and applying these recommendations is often required by the company standards and policies for cyber-security. They also play an essential role when the cybersecurity certification is required since the application of the recommendations has to be traced to the implementation in the system. VeriDevOps delivers important tools to:

- Locate the recommendations with semantic analysis of the requirements by ARQAN;
- Integrate cybersecurity tests into the CICD pipeline with RQCODE.

2.1. ARQAN and Semantic Mapping of Security Recommendations

Let us consider a typical scenario - a customer provided a document with a list of requirements. Some of those requirements concern cybersecurity. A developer has to locate those requirements in the text and address those requirements by following specific recommendations for a given domain.

For example, the customer specified a requirement stating - “The system has to provide secure password management.” For a customer, this requirement may appear to be of a sufficient level of detail. However, for a developer, it is quite puzzling, leading to a possible misinterpretation. If a developer tries to locate the recommendations in the STIG repository, unfortunately, the search engine will locate no recommendations.

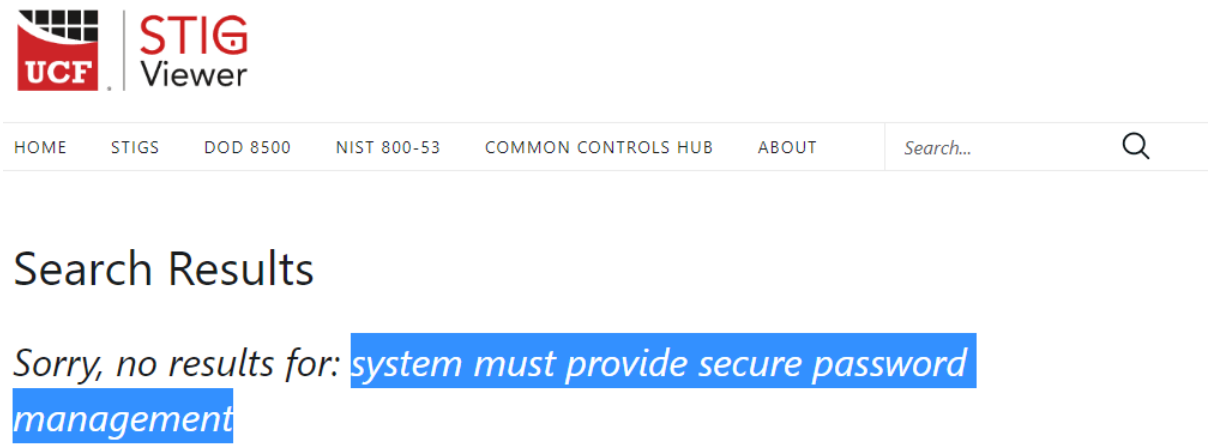


Figure 2. STIG Viewer[16], Empty search result for the requirement "System must provide secure password management"

ARQAN provides a semantic search in the STIG repository. Based on the analysis of the "meaning" of the requirements, ARQAN identifies semantically close recommendations. ARQAN applies a mechanism for vectorising requirements sentences and running cosine similarity with a whole set of vectorised STIG recommendations with the help of MPnet model by Microsoft [18].

Figure 3. ARQAN, Output of semantic search for the requirement "System must provide secure password management"

The resulting recommendations appear to be very relevant and, in particular, suggest implementing the password security by:

- Encrypting password storage and transmission;
- Requiring reset temporary passwords as soon as possible;
- Setting expiration dates;
- Limiting the number of times a password can be reused etc.

Many STIG recommendations come with indications for the methods to check that recommendations are taken into account or even with proposals for fixes. These methods are very practical, many times they include executable shell scripts. However, many times they are incomplete and have to be verified for correctness. In addition, maintaining a large set of scripts may be very time-consuming and inefficient. That is why we propose an approach that we call RQCODE that we outline below.

2.2. RQCODE and Security Requirements Patterns

STIG guidelines[16] provide thousands of security implementation recommendations for hundreds of systems. For example, in case an industrial PC runs on Ubuntu Linux distribution, STIG guidelines can suggest disabling a certain number of packages. For example STIG recommendation V_219157 states: *"Removing the Network Information Service (NIS) package decreases the risk of the accidental (or intentional) activation of NIS or NIS+ services."*. This recommendation suggests removing the NIS package from the system.

These STIGS often come with shell scripts for each recommendation for each package to be disabled. If two packages need to be disabled, two scripts need to be maintained. With each new version of Ubuntu (e.g versions 16, 18, 20) the number of scripts to maintain would double. If one needs to implement the same recommendation in CentOS, he or she will have to modify all scripts and change the Linux package manager e.g from apt to rpm.

We implemented these and many other recommendations as a SOOR in Java - the process that we called RQCODE or requirements as a code. While implementing the requirements and reversing them to UML with Modelio for analysis, we discovered a number of patterns. One of these patterns deals with disabling or enabling system packages. Separating the PackagePattern from the actual implementation of disabling particular packages helps to maintain the system of requirements, reuse it in other distributions and control complexity.

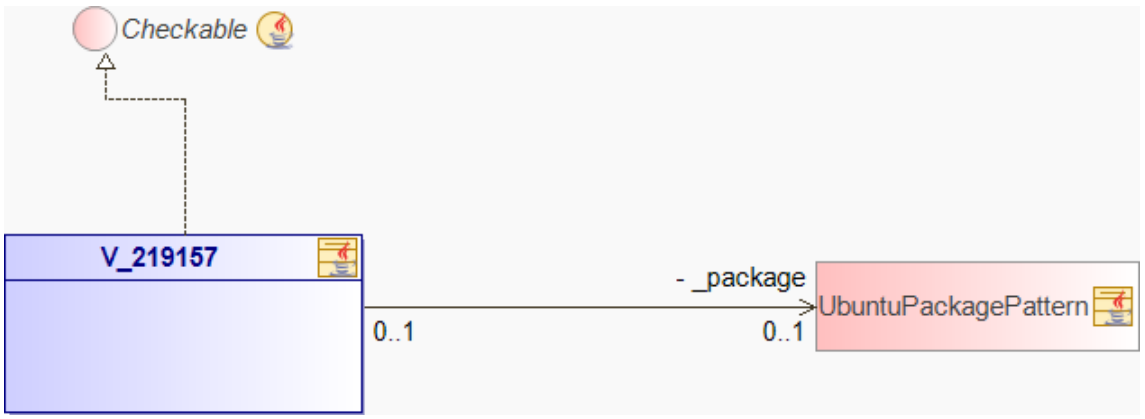


Figure 4. RQCODE UbuntuPackagePattern in UML

In case of PackagePattern it can be extended to CentOS distribution that uses a different package manager (rmp instead of apt) - changing one pattern class will help to run the same security recommendations in a new system. Thus, the approach improves efficiency and reduces a duplication in the important security related routines.

In addition, the use of RQCODE simplifies the matching of requirements with existing patterns for implementation. Moreover, the recommendations in RQCODE are compatible with the Continuous Integration and Delivery process since they can be directly applied in automated fashion - e.g., GitHub Actions.

2.3. Bridging Requirements and Tests

DevSecOps [11] promotes continuous attention to security aspects at development, delivery, and operation stages. This comes with reliance on automation everywhere where it is possible to. The existing tools address static analysis of the code, dependencies analysis, vulnerability scanning and monitoring. This all is integrated in a process at the organisational level to bring security aspects to the primary focus.

Most of the automation focuses on stages when the code is already available. Very few automation is proposed at inception and planning stages, while these stages are advocated to be crucial to gain efficiency in dealing with security. In VeriDevOps we suggest specific automation steps to provide developers with relevant security implementation guidelines in a timely manner. We discuss an illustrative example below.

We develop specific automation bots that implement several activities for requirements analysis. They may call ARQAN to classify a requirement or an *issue* in the GitHub repository as security related or not. In case, this issue is identified as one related to security, the bot may call ARQAN for security implementation recommendations semantically related to the requirement. When the recommendations are located in the STIG catalogue, the bot may look up into the RQCODE repository for the corresponding test and fixes classes. The developer is prompted to consult these recommendations and may import the corresponding tests and fixes in their own repository.

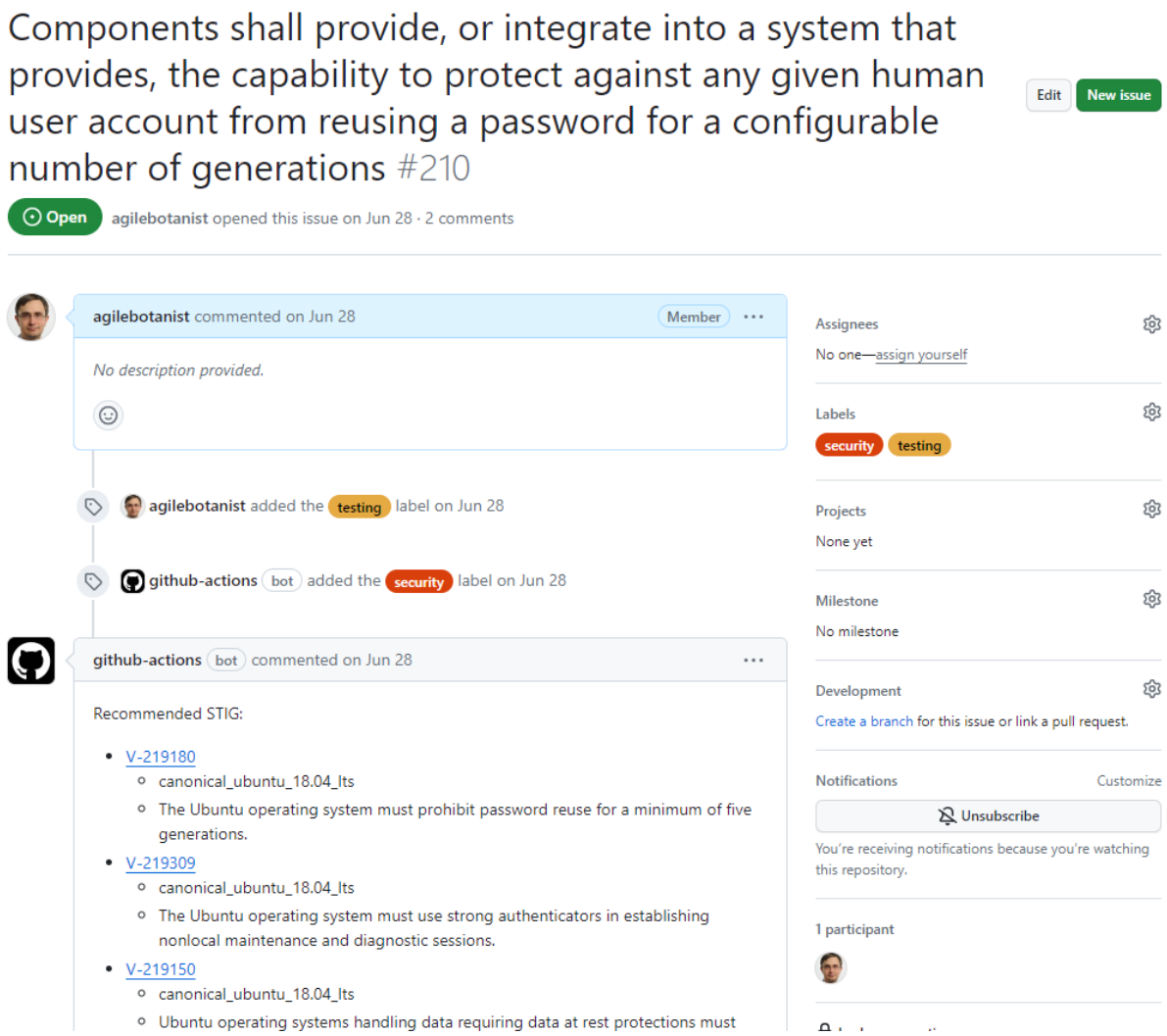


Figure 5. Illustration of Requirements Analysis Automation

In the figure above, a developer entered a requirement as an *issue* in the GitHub: *"Components shall provide, or integrate into a system that provides, the capability to protect against any given human user account from reusing a password for a configurable number of generations."*[1]
The GitHub Actions bot sent it to ARQAN and identified it as security related. After this action, the issue was labelled with a "security" sticker to notify the developer. Subsequently, the bot sent a request to ARQAN for a semantically relevant recommendation. The resulting recommendation, V-214961, was indicated in the comments for the issue. Next, the bot looked up in the RQCODE repository for a relevant test and fixes. The finding was indicated in the following comment.

3. Implementation of CI/CD Mechanisms

[1] This requirement comes from the IEC 62443 standard.

The above scenario is one of the possible automations for the security requirements analysis in a continuous process. Integration of these mapping mechanisms to the specific CI/CD infrastructure depending on the developers constraints.
We currently implemented two examples that are proposed in the following GitHub repositories:

- CICD ARQAN RQCODE with GitHub Actions [9] - implementation and a demonstrator for the CICD scenarios to automate security requirements analysis. This implementation fits for the open source projects fully relying on the GitHub infrastructure.
- CICD ARQAN RQCODE with Jenkins [10] - implementation and a demonstrator for the CICD scenarios to automate security requirements analysis. This implementation in an enabler to integrate of ARQAN and RQCODE in local or private environments, which is more suitable in an industry context.

Below, we outline the conceptual design of the GitHub Actions and Jenkins scenarios.

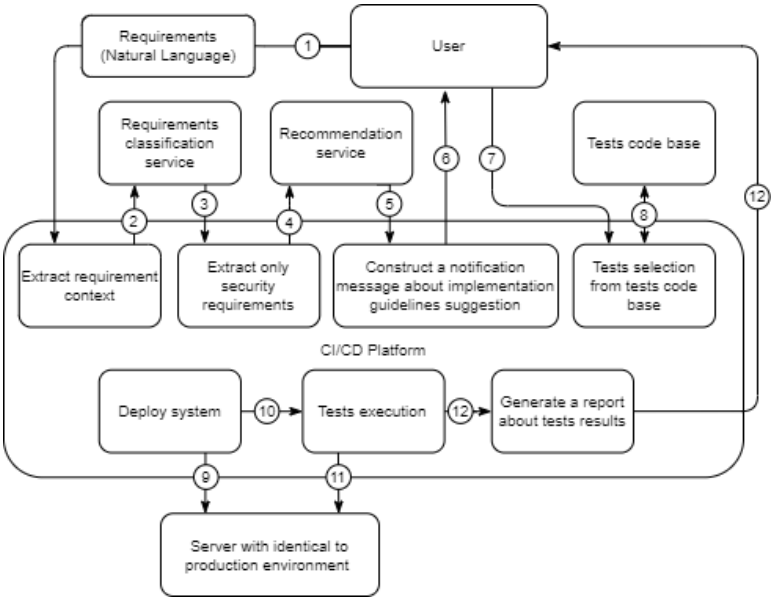


Figure 6. ARQAN and RQCODE in continuous analysis of requirements, guidelines mapping and tests execution.

As it is depicted in the figure above, (1) a user may express its requirements in Natural Language (NL) at any time. This requirement is then classified with ARQAN to be a security requirement or not (2)-(3). The recommendation service provides a mapping to STIG recommendations to the user (5)-(6). The user selects the relevant recommendations (7)-(8). This recommendation may have already been implemented as RQCODE classes (8). In this case, the RQCODE may be used to verify the pre-production system (9-11). The test reports are generated to inform the user of possible problems.
The figure below provides an overview of the above-mentioned approach implementation with GitHub Actions framework[2].

[2] https://github.com/features/actions

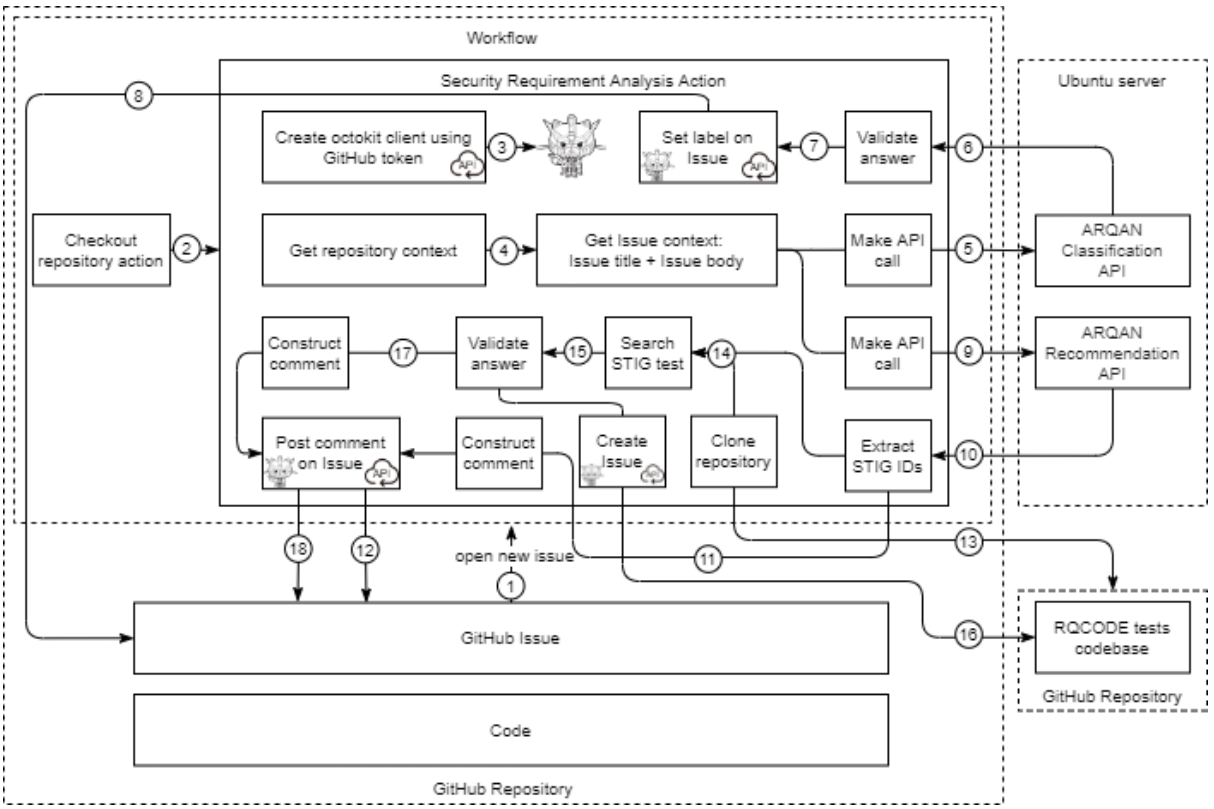


Figure 7. ARQAN + RQCODE integration scenario with GitHub Actions.

The user enters requirements as GitHub Issues. These issues are automatically classified to be related to security or not. For the security requirements, ARQAN locates relevant STIG rules and indicates them in the comments for the requirement. The user validates the relevant security guidelines. At the next step, the RQCODE repository is checked for STIGs rules implementation. If no implementation is located, the automation bot will create an issue in the RQCODE GitHub repository and will inform the developers.

4. Conclusions

In this document, we outlined the toolchain we developed to match security requirements with implementation recommendations that are instrumented with tests and fixes. The toolchain consists of ARQAN and RQCODE tool components that can be run in an automated scenario within CI/CD infrastructure. We have demonstrated these scenarios with specific implementation in GitHub Actions and Jenkins. The complete toolchain is fully available in VeriDevOps GitHub repository and supported by SOFTEAM.

We plan to further improve the approach in the following areas:

- Ranking approaches for search algorithms;
- Various vectorization and ML models for semantic search;
- Automation in applying RQCODE;

- User interface improvements in order to simplify the evaluation of the approach by the end-users.

The tools and the toolchain evolve to improve overall usability and help end-users to evaluate and adopt this approach.

References

[1] A. Sadovykh *et al.*, 'VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps', in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb. 2021, pp. 1330–1333. doi: 10.23919/DATE51398.2021.9474185.

[2] A. Ferrari, G. O. Spagnolo, and S. Gnesi, 'PURE: A Dataset of Public Requirements Documents', in *RE*, 2017, pp. 502–505.

[3] 'VeriDevOps. D2.3 Requirements automated generation tool chain'. [Online]. Available: https://www.veridevops.eu/

[4] 'VeriDevOps. D2.7. Patterns Catalogue', 2022. [Online]. Available: https://www.veridevops.eu/

[5] 'VeriDevOps. D1.6 VeriDevOps Framework - initial version'. [Online]. Available: https://www.veridevops.eu/

[6] A. Sadovykh *et al.*, 'VeriDevOps GitHub Repository'. 2022. Accessed: Jun. 30, 2022. [Online]. Available: https://github.com/VeriDevOps

[7] K. Iakovlev, A. Sadovykh, and A. Naumchev, 'ARQAN Services GitHub Repository'. 2022. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/VeriDevOps/ARQAN.services

[8] A. Naumchev, A. Sadovykh, and I. Nigmatulin, 'RQCODE for Windows 10 GitHub Repository'. Jan. 14, 2022. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/VeriDevOps/RQCODE/tree/master/src/main/java/rqcode/stigs/win10

[9] A. Sadovykh and A. Naumchev, 'CICD ARQAN RQCODE with GitHub Actions'. 2022. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/VeriDevOps/CICD_ARQAN_RQCODE_Github_Actions

[10] A. Sadovykh and A. Naumchev, 'CICD ARQAN RQCODE with Jenkins'. 2022. Accessed: Jan. 14, 2022. [Online]. Available: https://github.com/VeriDevOps/CICD_ARQAN_RQCODE_Jenkins

[11] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, 'Challenges and solutions when adopting DevSecOps: A systematic review', *Inf. Softw. Technol.*, vol. 141, p. 106700, Jan. 2022, doi: 10.1016/j.infsof.2021.106700.

[12] G. McGraw, 'Software security', *IEEE Secur. Priv.*, vol. 2, no. 2, pp. 80–83, 2004.

[13] M. Hafiz, P. Adamczyk, and R. E. Johnson, 'Organizing Security Patterns', *IEEE Softw.*, vol. 24, no. 4, pp. 52–60, Jul. 2007, doi: 10.1109/MS.2007.114.

[14] M. Gegick and L. Williams, 'On the design of more secure software-intensive systems by use of attack patterns', *Inf. Softw. Technol.*, vol. 49, no. 4, pp. 381–397, Apr. 2007, doi: 10.1016/j.infsof.2006.06.002.

[15] International Electrotechnical Commission, International Electrotechnical Commission, and Technical Committee 65, *Security for industrial automation and control systems. Part 2-4, Part 2-4,*. 2017.

[16] 'Security Technical Implementation Guide (STIG) Complete List', *STIG Viewer | Unified Compliance Framework®*. https://www.stigviewer.com/stigs (accessed Jan. 14, 2022).

[17] ‘OWASP Web Security Testing Guide | OWASP Foundation’. https://owasp.org/www-project-web-security-testing-guide/ (accessed Jan. 12, 2022).
[18] A. Hagen, ‘MPNet combines strengths of masked and permuted language modeling for language understanding’, *Microsoft Research*, Dec. 09, 2020. https://www.microsoft.com/en-us/research/blog/mpnet-combines-strengths-of-masked-and-permuted-language-modeling-for-language-understanding/ (accessed Sep. 29, 2022).