Smart Assistants for Enhancing System Security and Resilience

Andrey Sadovykh
Softeam
Paris, France
andrey.sadovykh@softeam.fr

Dragos Truscan, Tanwir Ahmad
Åbo Akademi University
Turku, Turku
firstname.lastname@abo.fi

Martin A. Schneider
Fraunhofer FOKUS
Berlin, Germany
martin.schneider@fokus.fraunhofer.de

Wissam Mallouli, Ana Cavalli
Montimage
Paris, France
firstname.lastname@montimage.org

Cristina Seceleanu
Mälardalen University
Västerås, Sweden
cristina.seceleanu@mdu.se

***Abstract*—Security and resilience have become paramount concerns for integrated system manufacturers as the number of vulnerabilities continues to increase annually. Cyber threats pose significant risks with substantial potential impacts on both manufacturers and end users. New regulations, such as the EU Cybersecurity Act and EU Cyber Resilience Act, mandate stricter practices and thorough verification throughout development and operations. Implementing a holistic DevSecOps process encompassing threat analysis, requirements engineering, development practices, verification, and operations management is challenging for large enterprises and SMEs. This complexity arises from the need for specialized expertise, knowledge of various techniques and tools, rigorous principle application, and thorough verification at each step, making the process costly, time-consuming, and potentially stifling to innovation and time-to-market.**

Our proposal introduces a suite of smart assistants designed to work collaboratively with engineers. These assistants recommend best practices and tools, suggest context-specific regulatory requirements, analyze design architecture, generate tailored code and configurations, and conduct resilience tests. This comprehensive approach aims to ensure the correctness and completeness necessary for security and regulatory compliance.

***Index Terms*—DevSecOps, Smart assistants, Security, Resilience, Requirements Engineering, Security by Design, Testing, Monitoring, Anomaly Detection**

I. INTRODUCTION

In the digital age, where software underpins virtually every aspect of modern life, security has emerged as a paramount concern [1], [2]. However, the relentless pursuit of fast deployment often takes precedence over robust security practices, leading to the proliferation of vulnerabilities and insecure applications [3]. This paper investigates the urgent need for a paradigm shift towards integrated hardware and software security engineering to address this pressing issue.

At the heart of this effort lies the recognition that software forms the backbone of IT infrastructures, services, and products. Yet, despite its pervasive influence, the current software development landscape prioritizes speed over security, leaving systems vulnerable to exploitation. Compounding this challenge is the fact that a significant portion of the software and hardware utilized within the European Union (EU) is developed outside its borders, necessitating stringent security requirements and their verification to align with EU standards.

Central to our investigation is the imperative for the EU to ensure the verifiability and auditability of software and hardware concerning their security. This includes a comprehensive analysis of the potential security implications associated with using open-source software and hardware, as well as strategies for enhancing security auditability within this context. The latest supply chain attacks on open source software projects such as xz utils backdoor [4] underpin this imperative.

Moreover, in a digital ecosystem characterized by perpetual updates and evolving regulatory frameworks, traditional approaches to security assessment fall short. Hence, there arises a critical need for methodologies and tools that facilitate continuous security assessments to adapt to the dynamic nature of modern software and hardware landscapes.

Several initiatives have proposed a holistic cybersecurity view under the DevOps paradigm. Among them, the VeriDevOps project [5] proposed integrating DevOps principles with early verification, test automation, and monitoring to ensure software security and reliability. It provides a systematic approach to embedding security requirements throughout the software development lifecycle. Key technologies include Natural Language Processing (NLP) for analyzing and formalizing security specifications and automated tools for quality assurance, system testing, and runtime monitoring. VeriDevOps automates the configuration of trace monitors based on security requirements and employs continuous monitoring to detect anomalies and vulnerabilities. It also generates attack tests to identify invalid states and potential security weaknesses. Additionally, it performs automated design and code checks using semi-structured and structured formalisms, either through model simulation or formal verification, to ensure compliance with security standards.

Thanks to AIDOaRt, grant agreement No 101007350 under JU ECSEL and Horizon 2020 funding programs

Applying VeriDevOps may pose several challenges. While it proposes more than 20 tools, it also requires expertise across multiple domains, including Threat Analysis, Security Requirements Engineering, various testing techniques, Monitoring, and Incident Analysis. Despite automating many steps, significant manual input is still necessary, making applying the methodology tedious. Additionally, integrating specific security methods for hardware development, operations, and verification into the workflow is complex. While VeriDevOps covers many scenarios, it does not encompass all possible security situations, necessitating further expansion to address diverse use cases comprehensively. Thus, applying VeriDevOps effectively demands both broad expertise and ongoing adaptation to cover more specific scenarios.

The proposal of this paper is to elevate the intelligence and automation of cybersecurity in software development by incorporating AI-based assistance that provides instructive support. This advanced assistance would provide recommendations for configuring and formalizing diverse security properties, selecting suitable testing techniques, and interpreting and elucidating the results. Additionally, it would offer boilerplates, examples, and detailed explanations for various methods and tools, thereby streamlining the implementation process.

One of the significant challenges in this approach is ensuring the AI-based system effectively suggests design approaches and coding practices that enhance system resilience, maintaining operation even amidst threats and attacks, including the hardware integration aspects. This necessitates the AI's ability to understand and apply resilience requirements accurately, guiding developers to meet these requirements and verify compliance effectively.

Furthermore, AI assistance must facilitate informed decision-making by recommending appropriate methods for satisfying specific resilience requirements. This includes providing insights into the application of these methods to ensure thorough verification of compliance. The challenge lies in the AI's capacity to interpret complex security specifications and translate them into actionable guidance that aligns with industry standards and best practices.

Overall, integrating AI-based assistance within VeriDevOps and enriching it with hardware aspects aims to overcome the challenges. By providing intelligent, context-aware recommendations and instructive support along with facilitating comprehensive verification processes, this approach seeks to enhance the reliability and security of software systems in a systematic and scalable manner.

The paper is structured as follows: Section II presents the background, including the achievement of the project and a review of related works, setting the context for SecDevOps and automation with smart assistants. Section III introduces the concept of employing smart assistants throughout different phases of the SecDevOps cycle, emphasizing its potential to enhance security practices, and outlines main scenarios and flows to illustrate the application of smart assistants in real-world development contexts. Finally, Section IV concludes the paper by summarizing key contributions and outlining future research and development avenues.

II. Background and Related Work

A. Cybersecurity Engineering Process

Cybersecurity implementation within a system must commence at the foundational level of requirements specification. These requirements guide system design, component selection, implementation, integration, and subsequent verification and validation processes. The efficacy of requirements specification hinges upon various parameters, notably clarity, atomicity, and verifiability. Formal and semi-formal specification techniques play a pivotal role in automating the verification process, thereby aiding in identifying and mitigating potential cybersecurity threats throughout the system's lifecycle. Initiating with a comprehensive understanding of the system's scope and the assets requiring protection against cybersecurity threats, the requirements specification informs critical design decisions. These decisions encompass the selection of appropriate hardware and software platforms, as well as the configuration of the application stack. Moreover, the design phase necessitates strategic considerations regarding access control mechanisms and internal restrictions, which, in turn, dictate the architectural blueprint of the implemented system. After design, adherence to specific coding practices becomes imperative to minimize the injection of vulnerabilities and fortify mechanisms for secure storage and access of sensitive information.

Beyond the developmental phase, systems are inevitably exposed to many attacks aiming at exploiting vulnerabilities within the application, software, and hardware infrastructure. Given the omnipresent nature of vulnerabilities, incessantly discovered at a mass scale, preemptive measures must be in place to detect and manage these vulnerabilities through timely patches and other protective measures. Anticipating vulnerabilities before disclosure underscores the importance of anomaly detection mechanisms capable of preemptively identifying potential attack vectors.

Moreover, hardware resilience assessment is essential in ensuring the overall security and reliability of computing systems, as it complements software resilience assessment by addressing vulnerabilities inherent in the physical components. Techniques for hardware resilience assessment include rigorous testing for fault tolerance, stress testing, and examining supply chain integrity. These assessments face significant challenges, such as detecting and mitigating hardware backdoors, counterfeit components, and vulnerabilities introduced during manufacturing. Given that hardware can originate from untrusted sources or countries, it is critical to validate its security through thorough inspection and verification processes. This layered approach helps prevent the software from inheriting hardware-related vulnerabilities, thereby fortifying the entire system's resilience.

Resilience in software and hardware systems is fundamentally the ability of a system to resist, absorb, recover from, and adapt to adverse conditions, particularly in the face of physical

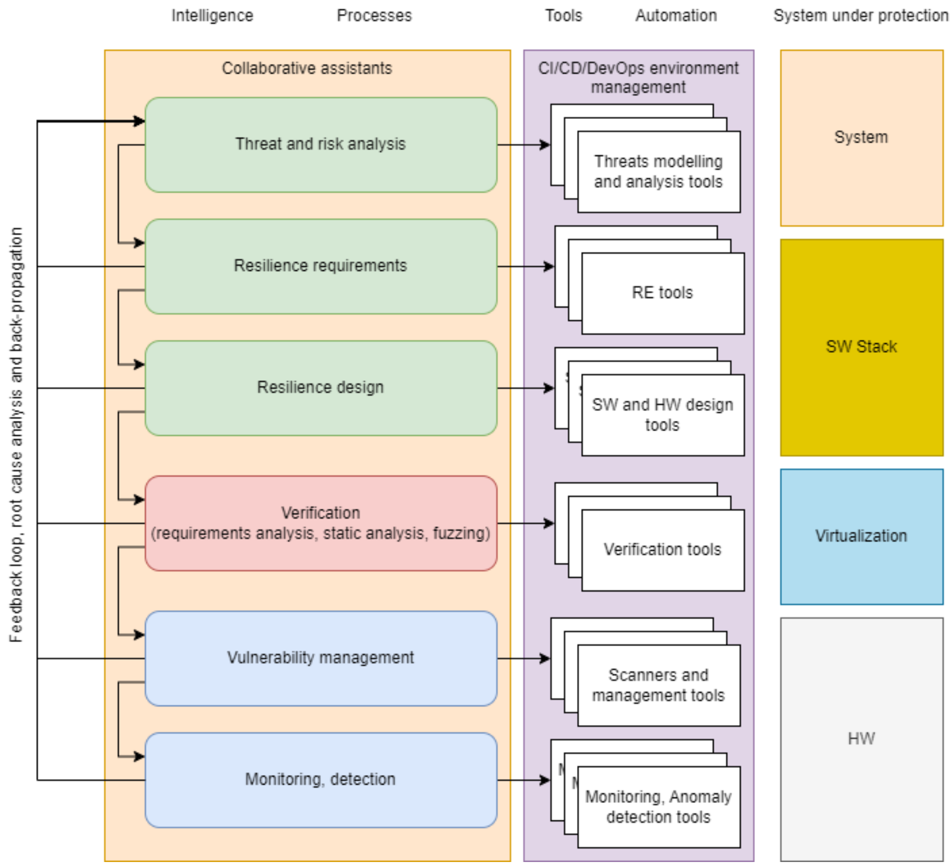

Fig. 1. The concept of Smart Assistants for Continuous Holistic Security Verification

and sophisticated cyber-attacks [6]. These systems, characterized by the integration of physical components and software elements, are inherently susceptible to a range of threats aimed at disrupting system availability, perturbing performance, and other malign objectives. The repercussions of such attacks extend beyond system disruption and can trigger multilevel consequences, including economic, social, and environmental impacts. Moreover, the interconnected nature of their components means that an attack on one part of the system can negatively affect other parts. This interconnectedness is even more pronounced in our world of interlinked critical infrastructures, where an attack on one system can trigger cascading failures across others. Therefore, it is imperative that these systems are engineered with resilience in mind, adopting proactive designs and reactive countermeasures to effectively mitigate these threats [7]. These resilience techniques are multifaceted and can be classified into several categories as proposed by NIST[1].

B. The VeriDevOps Framework

VeriDevOps project proposed a methodology [8] that merges DevOps principles with early verification, test automation, and monitoring, ensuring the security and reliability of software systems. It provides a structured approach to software development, emphasizing the continuous integration of security requirements throughout the development lifecycle.

At its core, VeriDevOps automates key aspects of software development with security in mind. This includes defining and analyzing security requirements, conducting system testing and monitoring, and integrating these processes into established VeriDevOps practices.

The process begins with analyzing and formalizing text-based security requirements gathered from various sources. Natural Language Processing (NLP) and pattern recognition technologies play a crucial role in maintaining consistency and clarity in these specifications. Additionally, patterns are translated into temporal logic for better understanding.

Another essential aspect is the automated configuration of trace monitors, which are based on formalized security requirements using structured formalisms. These monitors are continuously adjusted and monitored over time to detect anomalies and vulnerabilities during runtime.

Furthermore, VeriDevOps automatically generates attack tests based on security requirements, aiming to expose potential vulnerabilities by pushing the system into invalid states. These tests complement positive testing methods and reveal insecure behaviors that may go unnoticed otherwise. To further improve testing, guidelines can be established for testers to propose scenarios that evaluate both security and energy

[1]The NIST Directive 800-160: https://csrc.nist.gov/pubs/sp/800/160/v2/r1/final

properties, often overlooked areas.

Finally, VeriDevOps automates design and code checks according to specified security requirements using semi-structured and structured formalisms. These verification activities can be carried out through simulation or formal verification of system descriptions.

The VeriDevOps Methodology (Fig. 2) encompasses a suite of interconnected tool sets designed for Security Requirements Generation, Reactive Protection at Runtime, and Prevention at Design and Development. These tool sets are closely integrated to align security requirements with design analysis, code-level verification, and runtime system analysis. They comprise concrete tool components provided and developed by VeriDevOps partners, varying in licensing policies and maturity levels. While some tools are well-established commercial or open-source solutions, others are more experimental. However, all tools must adhere to the interfaces and features outlined in the VeriDevOps Methodology and be interchangeable to a certain extent. Case studies combine these tools in a specific industry context based on their alignment with requirements and compatibility with industry practices.

1) Requirements Specification: In the VeriDevOps process, security requirements undergo examination and formalization, sourced from diverse textual descriptions. To ensure consistency and clarity while avoiding inconsistencies and ambiguities, we leverage Natural Language Processing (NLP) alongside established patterns or boilerplates. Additionally, techniques are employed to automatically translate these patterns into temporal logic, further enhancing requirement clarity and consistency. Various techniques, such as PROPAS and RQCODE, can be integrated into the VeriDevOps methodology for requirements formalization. Manual and semi-automatic translation methods are also employed to optimize this process. Furthermore, verification and analysis tasks can be executed by either simulating the final model or verifying the system's description. Using natural languages and model smells, we've established indicators (e.g., NALABS) for security requirement flaws and defined metrics to automatically detect these flaws in security artifacts.

2) Prevention at Development: During this phase, multiple techniques are employed for test modeling (e.g., UPPAAL, PyLC, Modelio, GW2UPPAAL), automated test generation (e.g., MetaTester, CompleteTest, Graphwalker), and vulnerability localization (e.g., Localizer, RCA). This information aids in generating both positive and negative tests intended to push the system into specific states, thereby exposing potential vulnerabilities. To enhance this process, establishing guidelines and a format enabling testers to design scenarios evaluating not only security aspects but also energy properties would be beneficial, as energy properties are often overlooked in testing.

3) Protection at Operations: An automated setup of monitoring tools (e.g., MMT, THOE, EARLY), based on the specification of security requirements in natural language, semi-structured, or structured formalisms is available. Over time, these traces are automatically configured and continuously observed using formal or semi-formal specifications. Runtime monitoring, which observes system behavior during operation, is implemented to detect errors, monitor performance, ensure compliance, and maintain system health. This serves as a foundation for potential preemptive countermeasures.

VeriDevOps represents a departure from traditional software development methodologies by placing security at the forefront of the development process. By integrating security requirements analysis, automated testing, vulnerability localization, and continuous monitoring into the DevOps pipeline, VeriDevOps ensures a holistic approach to software security. This innovative integration enables organizations to proactively identify and mitigate security threats throughout the software development lifecycle, thereby enhancing the resilience of software systems.

C. Challenges

Addressing security challenges within the DevOps framework involves navigating several complex issues. Firstly, integrating security requirements into the DevOps pipeline presents a significant challenge. This requires ensuring that security considerations are seamlessly woven into the existing development and deployment processes without impeding agility or efficiency.

Secondly, achieving clarity and detailing security requirements is crucial yet challenging. It necessitates expressing security needs in a clear, unambiguous manner that leaves no room for interpretation. However, achieving this clarity becomes more complex when utilizing diverse formal methods and tools across different stages of the development lifecycle.

Moreover, integrating security requirements analysis and verification throughout the entire DevOps process is essential but challenging. It involves overcoming barriers to incorporating security considerations at every stage, ensuring consistency and accuracy in specifying and analyzing requirements across diverse environments and toolsets.

Additionally, automating security test generation and selection poses its own set of challenges. Identifying appropriate tools and techniques for generating automated security tests and ensuring their seamless integration within the DevOps pipeline can be a daunting task, requiring careful consideration of compatibility and effectiveness.

Furthermore, implementing robust security monitoring systems presents another significant challenge. This entails establishing comprehensive monitoring across all critical components of the DevOps pipeline, detecting and responding to security threats in real-time while maintaining system performance and reliability.

Another significant set of challenges involves supporting developers and guiding them through the implementation of the DevSecOps methodology. This includes facilitating their ability to select appropriate methods and tools, configure them effectively, and utilize them proficiently within the DevSecOps framework. These challenges encompass various aspects such as formal specifications, static analysis, testing methodologies, monitoring systems, root cause analysis techniques, and the

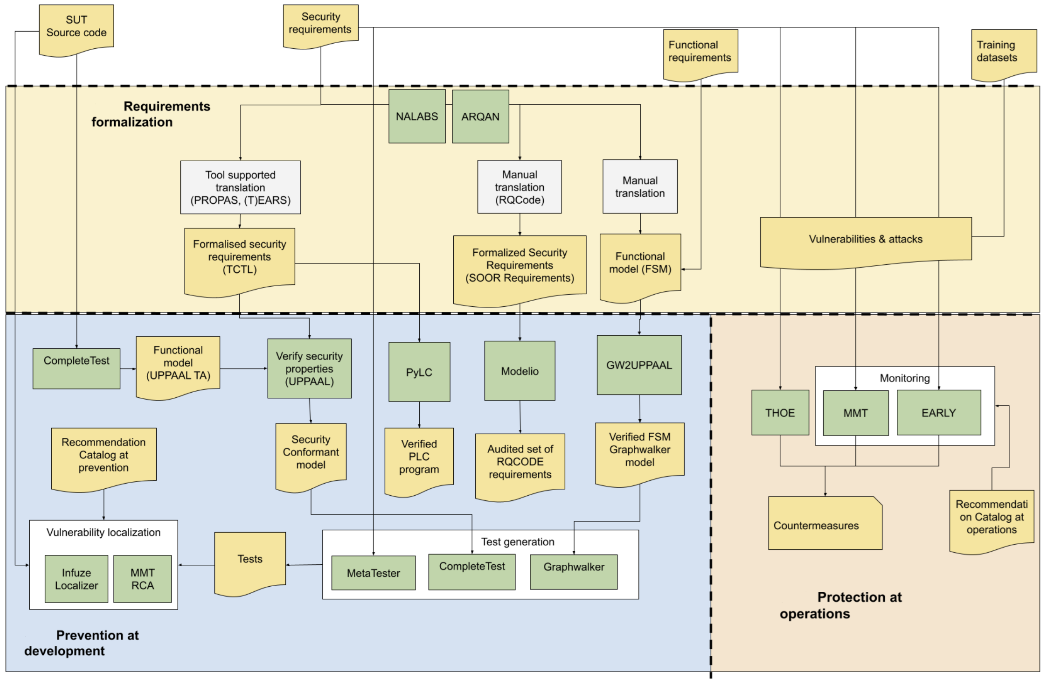

Fig. 2. The VeriDevOps Framework

management of vulnerabilities and rework. Addressing these challenges is essential for empowering developers to seamlessly integrate security practices into the software development lifecycle while ensuring the reliability and security of the resulting software products.

Finally, integrating methodologies, tools, and assistant technologies within Continuous Integration, Deployment, DevOps and CyberOps practices is indispensable. By embedding security checks within automated pipelines and establishing real-time monitoring mechanisms, organizations can ensure adherence to quality standards and resilience practices across the system lifecycle's developmental and operational phases. This holistic approach enhances system security and fosters a culture of proactive cybersecurity within the organizational framework.

D. Smart Assistants Background

The use of smart assistants (sometimes termed as *bots*) in developing computing systems has become increasingly prominent, primarily due to advancements in generative artificial intelligence (AI) and machine learning (ML). Smart assistants in the context of software engineering are tools and platforms that leverage AI and ML to aid developers in various aspects of software development, maintenance, and management. These AI-driven tools enhance productivity, improve code quality, and streamline development processes.

The integration of smart assistants into software engineering is transforming the way developers write code, test software, manage projects, and interact with development environments. As AI technologies continue to evolve, these tools are expected to become even more sophisticated, further enhancing their capabilities to support the software development lifecycle.

The penetration of smart assistants has been observed in all major areas of the software and system development process to assist with different tasks. As summarized by several systematic literature reviews on the topic [9]–[11] the assistants can span tasks from *development automation* (requirements processing, code generation [12], debugging, testing [13], documentation generation), *real-time collaboration and support* (coding assistance via code completion [14], error detection and correction [15], [16], code reviews [17]–[19]) to *project management* (tracking progress, predicting timelines, and identifying bottlenecks) just to enummerate the main ones.

Of particular importance, the evolution of large language models (LLMs) has enabled many of the previously enumerated activities. However, recent studies [20] have shown that the benefits of LLMs are limited by a set of open problems such as hallucinations and the importance of complementing the former with complementary techniques. Nevertheless, the authors of [21] provide a survey of how LLM-based agents can support the planning activities of complex processes, which can also be applied to IT systems.

Following similar techniques and practices as described above, smart assistants can also be used to assist with cybersecurity-related activities throughout the software and system development process. Threat detection and analysis, automated response, vulnerability management, compliance and reporting, security training and awareness, and incident forensics and analysis are just a few of the areas that can benefit from the capabilities of smart assistants. However, as emphasised in [22], there is still a need for automated intelligent tools to assist cybersecurity-related tasks.

III. Smart Assistants for Continuous Holistic Security Verification

Utilizing AI-driven smart assistants within the VeriDevOps framework represents a cutting-edge innovation in software development. These assistants facilitate various aspects of the security lifecycle, from requirements specification to vulnerability analysis and remediation. By harnessing the power of AI, VeriDevOps empowers development teams to make informed security decisions and streamline security-related tasks, ultimately improving the efficiency and effectiveness of the development process.

Integrating smart assistants within the VeriDevOps framework offers a multifaceted approach to addressing cybersecurity challenges across the software development lifecycle. Here, we outline key concepts and provide illustrative scenarios to motivate the application of smart assistants in real-world development.

To address the needs outlined above, we propose the comprehensive integration of specialized smart assistants across all development lifecycle phases, including threat and risk analysis, verification, and monitoring, as depicted in Figure 1. These smart assistants will collaborate and assume responsibilities such as selecting and configuring task-specific tools, generating relevant artefacts, and evaluating the outcomes of these processes. For instance, a smart assistant designated for verification tasks might select and configure appropriate tools for verification, such as static analysis and fuzzing tools, execute the analysis and testing procedures, and assess the results, as depicted in Figure 3.

To ensure a comprehensive approach towards the development lifecycle, the scope of the smart assistants extends beyond individual development phases and activities. These assistants will also facilitate exchanges and collaboration among various smart assistants responsible for different tasks. This is achieved through the sharing of results, thereby creating a continuous feedback loop among the smart assistants. As demonstrated in Figure 3, this systematic exchange of information fosters a holistic understanding of the cybersecurity posture of a product. Other smart assistants can leverage this aggregated information to enhance their outputs, for instance, by incorporating identified threats and risks into the verification phase.

Additionally, this collaborative framework benefits the operational dynamics among smart assistants and provides manufacturers with a comprehensive overview of the system. This is achieved through the storage and further processing of artefacts within a knowledge base, which is then utilized to generate summaries and overviews of various activities, thus offering a consolidated view of system security and performance.

A. *Smart Assistants for Structured Resilience Requirements*

Security presents a dynamic challenge shaped by an ever-evolving threat landscape and shifting regulatory frameworks. Recent sophisticated attempts to compromise the Linux kernel via supply chain attacks using xz utils, coupled with the enactment of the Cyber Security Act (CSA) and the Cyber Resilience Act (CRA), highlight the imperative for refined development methodologies. These methodologies should not only enable engineers to create and deliver products with elevated security standards and thorough verification and validation but also assist them in comprehending and integrating regulatory demands and requirements into the development process to demonstrate compliance with pertinent regulations and standards, such as IEC 62443 and ETSI EN 603645, and potentially certification frameworks like the new EUCC. Furthermore, emerging threats and risks associated with the supply chain necessitate a deeper understanding to address and document these concerns effectively.

Central to our proposal is AI-based smart assistants designed to optimize cyber resilience through enhanced threat modelling and analysis. These smart assistants utilize established standards like IEC 62443, emerging regulations such as the CSA and CRA, and NIST guidelines to identify, analyze, and understand the implications of threats. Acting as a pivotal interface between system engineers and the development processes, these smart assistants ingest threat data from diverse sources, including vulnerability databases such as Common Vulnerabilities and Exposures (CVEs), Common Weakness Enumerations (CWEs), Security Technical Implementation Guides (STIGs), Threat Intelligence and Management Platforms, and newly identified vulnerabilities (0-day vulnerabilities). By referencing industry standards and regulations, the smart assistants evaluate the relevance of these threats to the system under development, ensuring compliance and pinpointing specific threats that need mitigation. This facilitates a comprehensive threat modelling process, aligning analyzed threat data with the system's design specifications, thereby enabling system engineers to proactively address vulnerabilities and incorporate security measures from the project's inception.

Due to the increasing complexity of security threats, effective yet flexible specification methods that support rigorous analysis of software security requirements are needed. Security requirements specifications that consider thematic roles and domain knowledge to enable deep semantic analysis are desirable. We aim to develop specific assistants similar to those for code generation (e.g., GitHub Copilot[2]), which will empower engineers to generate consistent and testable specifications by interpreting natural language security requirements or partial code inputs. The assistant will be based on our work on the semantic analysis framework of *ReSA*, a structured, pattern-based language and ontology for specifying embedded systems requirements [23]–[25], as well as on our previous work on EARS-based test generation for PLC programs [26].

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

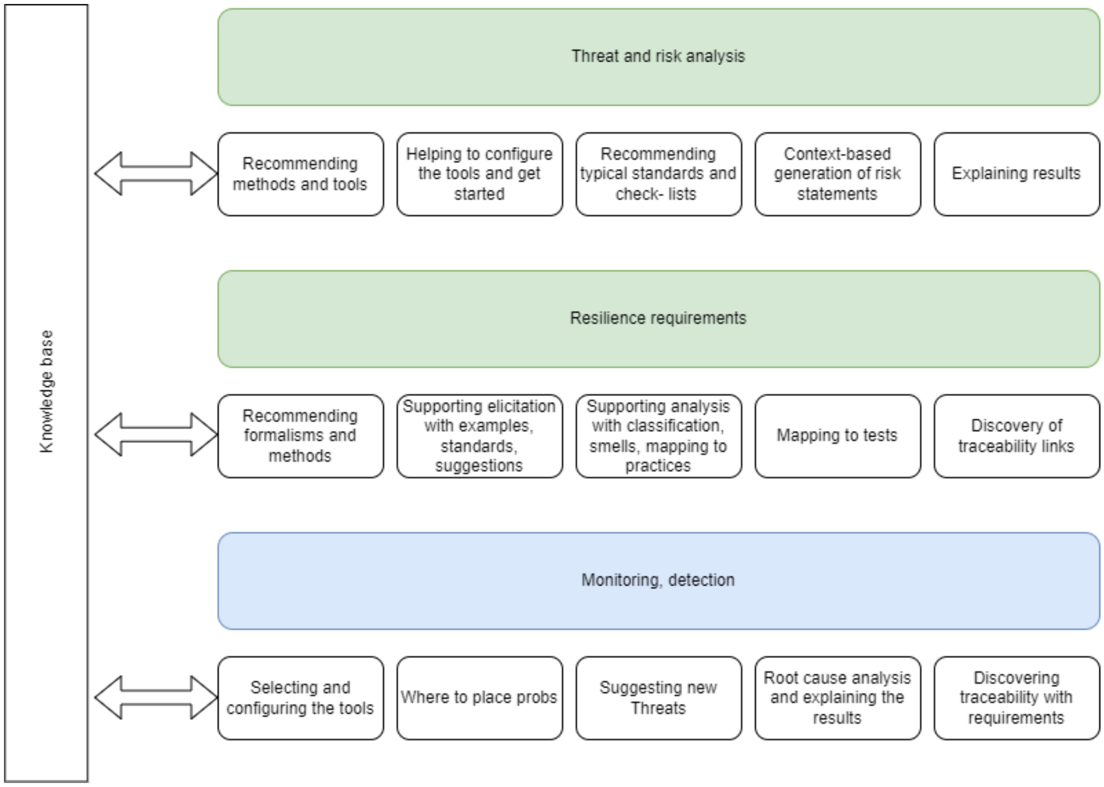

Fig. 3. Tasks examples supported by Smart Assistants

1) Example: Automated Threat Modelling and Analysis Flow: **Scenario:** A multinational manufacturing company is upgrading its industrial control systems (ICS) to enhance security and comply with international standards and regulations. The company aims to ensure that its systems are resilient against cyber threats and comply with relevant regulatory requirements.

Flow: To utilize smart assistants for conducting resilience threats analysis and suggesting mandatory and recommended requirements from EU regulatory frameworks (such as the EU Cyber Resilience Act), IEC 62443, NIST guidelines, and the EU Common Criteria.

1) *Resilience Properties Modelling:* The company's cybersecurity team initiates the resilience threats analysis process for the new ICS upgrade project. They model system properties with the smart assistant, including the system's architecture, intended operational environment, and potential threat vectors.
2) *Threat Identification:* The smart assistant analyzes the properties and identifies potential resilience threats specific to the ICS, such as supply chain attacks, insider threats, and vulnerabilities in communication protocols.
3) *Regulatory and Standards Compliance:* The smart assistant aligns identified threats with regulatory standards, suggesting mandatory requirements from the EU Cyber Resilience Act, guidelines from IEC 62443, best practices from NIST, and security assurance methods from the EU Common Criteria, covering risk management, incident reporting, access control, monitoring, and evaluation.
4) *Resilience Requirements Suggestions:* The smart assistant provides a categorized list of mandatory and recommended requirements, such as risk management (EU Cyber Resilience Act), Security Level 3 compliance (IEC 62443), incident response planning (NIST), and EAL4+ certification (EU Common Criteria), along with recommendations like continuous monitoring, security audits, penetration testing, and secure SDLC practices.
5) *Resilience Planning, Implementation and Verification:* The cybersecurity team reviews the smart assistant's suggestions and develops an action plan, assigning responsibilities, setting timelines, and allocating resources for each compliance activity. The team implements the mandatory and recommended requirements, using the smart assistant for guidance. The assistant continuously monitors compliance status and alerts the team to deviations or emerging threats. After implementation, the smart assistant helps verify compliance through automated checks and generates detailed reports. These reports are used for internal review and are submitted to regulatory bodies for compliance verification.

B. Smart Assistants for Security Testing

Cybersecurity testing represents a critical phase in the software development lifecycle, comprising various activities including strategic planning, risk and requirements analysis, test design, execution, evaluation, and comprehensive reporting. There exists a plethora of methodologies, strategies, and tools designed to facilitate these processes. Notwithstanding, activities such as planning and requirements analysis frequently remain manual tasks. Smart assistants can be invaluable in these areas, particularly in translating test strategies, cybersecurity requirements, and risk assessments into detailed test plans. Additionally, smart assistants can streamline tasks related to the preparation of testing processes, such as the selection and configuration of tools for different methodologies, which can be an arduous activity. Deciphering the outcomes of cybersecurity testing requires extensive technical knowledge concerning the System Under Test (SUT), its operating platform, the programming languages utilized, and weaknesses to accurately evaluate the implications of vulnerabilities. Moreover, techniques like static analysis and dynamic testing, for instance, fuzzing, exhibit unique strengths and limitations necessitating further analysis and additional testing to refine and verify results. In this context, smart assistants can play a pivotal role in verifying and refining results by deploying advanced tools to convert intermediate data into final outcomes, effectively distinguishing between true and false positives, thereby improving the accuracy of cybersecurity testing. Ultimately, smart assistants contribute to the development of effective security patches for detected vulnerabilities through automated program repair techniques, e.g., using the application of Code Large Language Models (LLMs).

1) Example: Security Test Generation Flow: **Scenario**: A software development team is tasked with building a new web application with stringent security requirements. They leverage automated test generation and vulnerability localization tools within the VeriDevOps framework to ensure the application's resilience against potential cyber threats.

Flow:

1) *Security Requirements Analysis*: The development team is using smart assistants to identify comprehensive security test requirements for the web application based on the common knowledge base populated by the smart assistant.
2) *Security Test Planning Control*: Smart assistant will support the planning of the security testing, in particular propose complementary security testing tools and test exit criteria based on the security test requirements, existing licenses, used techology.
3) *Security Test Generation Execution*: The development team selects from the security testing tools proposed by the smart assistants in the planning phase those that fit best to their experience. Based on the employed technology, the smart assistants configure the selected security testing tools, using knowledge from previous projects and from the community. The development team checks the configuration and makes modification where necessary based on their own expertise. Smart assistants learn from these modification for future processes. Using security test generation tools configured by smart assistants and the development in a collaboratavi manner, a test suite is generated and executed, aimed at evaluating the application's security posture. The test suite encompass both positive and negative scenarios, covering various attack vectors and potential vulnerabilities.
4) *Security Test Evaluation* The test cases produce a large number of a variety of results. Smart assistants help the development team to get an overview, through evaluation of the test results, e.g., by analyzing them with respect to uniques, relevance, and severity. Further tests can be generated and executed by smart assistants to obtain more information on findings. Finally, smart assistant can automatically populate the bug tracking system with consolidated information from the test evaluation, prioritize them based on test results and threat analysis results, and alert the development teams if urgent action is required.
5) *Reporting*: Smart assistants generate based draft test reports based on the performed activities based on the requirements from authorities they have been fed with. The development team completes, reviews and finalize the test reports. Smart assistants can also summarize these reports for the project management.
6) *Remediation and Patching*: The development team addresses the identified vulnerabilities. Smart assistants propose potential securuty patches and assess their appropriateness using patch validation techniques, e.g., generating further test cases. The development team selects promising patch candidates, improves them. Smart assistants perform again patch validation and regression techniques to assess the patch until an effective patch has been identified.

This flow demonstrates how the development team can benefit from smart assistants in all the stages of security testing to prepare, perform, evaluate and report security tests and their results in a consolidated and efficient manner, where the development teams and smart assistants are working together hand in hand, to identify and fix vulnerabilities with increased efficiency.

C. Smart Assistants for Resilience Monitoring

Specific assistants have to be developed to help monitor, detect, and respond to security threats more efficiently. By enhancing and automating many aspects of cybersecurity operations, smart assistants are becoming an essential component of modern security strategies, helping to mitigate the increasing complexity and frequency of cyber threats. Smart assistants will enable Automated Threat Detection by continuously monitoring network traffic, system logs, and other data sources for unusual or suspicious activity and detecting anomalies that might indicate a breach or an attempted attack. Upon detecting a potential threat, these AI-driven systems can generate real-time alerts. This immediate notification allows security teams to act swiftly, potentially stopping attackers before they can cause significant damage. As such, smart assistants will provide automated response capabilities, deciding and executing predefined actions when certain types of threats are detected. This might include isolating affected systems, blocking IP addresses, or initiating patches to vulnerable software. In addition, by analyzing historical data and identifying trends, smart assistants can predict and identify potential future threats. This predictive capability helps in proactive threat management, allowing organizations to strengthen defenses before an attack occurs.

1) Example: Automated Threat Detection and Response Flow: **Scenario**: An organization is deploying a new software application that handles sensitive user data. To ensure the security of this application, the organization integrates automated threat detection and response mechanisms by applying the VeriDevOps pipeline with a help of specific assistants.

Flow:

1) *Security Requirements Specification*: The organization generate structured security requirements based on natural language descriptions of potential threats and vulnerabilities associated with the application.
2) *Continuous Monitoring Setup*: Smart assistants configure monitoring tools to continuously monitor network traffic, system logs, and application behaviour for suspicious activity.
3) *Real-time Threat Detection*: As the application is deployed and operational, the monitoring tools detect anomalous patterns in user access patterns and data usage, signalling potential security threats.
4) *Automated Alert Generation*: Upon detecting suspicious activity, the monitoring tools automatically generate real-time alerts, notifying the security team of the potential security breach.
5) *Automated Response*: Smart assistants trigger predefined automated response actions, such as isolating affected systems, blocking IP addresses associated with suspicious activity, and initiating patches to mitigate vulnerabilities.
6) *Incident Analysis and Resolution*: The security team analyzes the alerts and response actions, investigating the root cause of the security incident and implementing further measures to prevent similar incidents in the future.

This flow illustrates how automated threat detection and response mechanisms, integrated within the VeriDevOps framework, enable organizations to identify and mitigate security threats in real-time proactively.

D. Smart Assistants Collaboration

Our proposed framework leverages a hierarchical structure of smart assistant agents to address various aspects of the DevSecOps lifecycle. High-level agents oversee the entire process, strategically delegating tasks to lower-level agents specializing in specific areas. These lower-level agents can automate tasks such as identifying vulnerabilities based on established security protocols, performing code reviews in order to detect potential security concerns within the codebase, generating test cases, and even proposing patches for vulnerabilities. This collaborative approach fosters a more efficient workflow, where specialized agents handle routine tasks, while high-level agents maintain an overarching view and ensure progress towards overall security objectives.

To further enhance the reliability of the framework, the agents can employ self-assessment techniques. These techniques involve cross-referencing findings with established security protocols and identifying potential biases or errors within the generated outputs, such as e.g. ”hallucinations” in the context of LLMs. When necessary, the agents can seek clarification from human experts, ensuring the accuracy and effectiveness of their work. This self-assessment helps to minimize errors and ensures that the agents operate within the bounds of established security best practices.

In summary, the proposed framework utilizing a hierarchy of smart assistants represents a significant advancement towards a more automated, efficient, and secure software development lifecycle. By automating and streamlining various tasks, developers and operations teams can focus their efforts on more complex and critical aspects of the development lifecycle. Furthermore, we plan to have proper safeguards, validation mechanisms, and human oversight within the framework to ensure the reliability and security of the outputs generated by these agents. Additionally, continuous training and refinement of AI-based assistants would be necessary to keep pace with the ever-evolving landscape of software development practices and security threats.

IV. Conclusions

In this paper, we have presented a proposal for a holistic approach to enhancing software security and resilience through the integration of smart assistants within the software development lifecycle. By combining innovative methodologies, automated processes, and AI-driven assistance, our approach offers a comprehensive framework for building secure and resilient software systems.

We introduced the concept of smart assistants tailored to various stages of the software development lifecycle, from requirements specification to vulnerability analysis and remediation. These assistants leverage AI technologies to empower development teams, streamline security-related tasks, and make informed security decisions.

Through the integration of smart assistants, organizations can proactively identify and mitigate security threats throughout the development process. By automating tasks such as security requirements analysis, code review, and vulnerability testing, smart assistants enable development teams to focus on building high-quality software while ensuring security best practices are followed.

However, the adoption of smart assistants is not without challenges. Integration complexities, ensuring clarity of security requirements, and addressing limitations of automation are among the hurdles that organizations may encounter. Overcoming these challenges will require collaboration across the organization and ongoing research and development efforts in the field of software security.

Despite these challenges, the potential benefits of integrating smart assistants into the software development lifecycle are significant. By leveraging AI-driven assistance, organizations can build more secure and resilient software products, ultimately mitigating the impact of cyber threats and protecting critical digital assets.

In conclusion, the integration of smart assistants represents a promising approach to enhancing software security in today's digital landscape. By embracing innovation, addressing challenges, and fostering collaboration, organizations can leverage smart assistants to build a more secure and resilient digital future.

Acknowledgment

The proposed work is the fruit of the authors' extensive collaboration in many research projects and mainly AIDOaRt [27], which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, and Spain.

References

[1] M. Cankar, N. Petrovic, J. P. Costa, A. Cernivec, J. Antic, T. Martincic, and D. Stepec, "Security in devsecops: Applying tools and machine learning to verification and monitoring steps," in *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, ICPE 2023, Coimbra, Portugal, April 15-19, 2023* (M. Vieira, V. Cardellini, A. D. Marco, and P. Tuma, eds.), pp. 201–205, ACM, 2023.

[2] Y. He, G. Meng, K. Chen, X. Hu, and J. He, "Towards security threats of deep learning systems: A survey," *IEEE Trans. Software Eng.*, vol. 48, no. 5, pp. 1743–1770, 2022.

[3] D. Bassi and H. Singh, "A systematic literature review on software vulnerability prediction models," *IEEE Access*, vol. 11, pp. 110289–110311, 2023.

[4] A. Freund, "backdoor in upstream xz/liblzma leading to ssh server compromise. post on mailing list oss-security@openwall.." https://openwall.com/lists/oss-security/2024/03/29/4, 2024. Accessed: 2024-04-25.

[5] A. Sadovykh, G. Widforss, D. Truscan, E. P. Enoiu, W. Mallouli, R. Iglesias, A. Bagnto, and O. Hendel, "VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb. 2021.

[6] J. Jia, Y. Liu, G. Zhang, Y. Gao, and D. Qian, "Software approaches for resilience of high performance computing systems: a survey," *Frontiers Comput. Sci.*, vol. 17, no. 4, p. 174105, 2023.

[7] S. M. Alhidaifi, M. R. Asghar, and I. S. Ansari, "A survey on cyber resilience: Key strategies, research challenges, and future directions," *ACM Comput. Surv.*, vol. 56, no. 8, pp. 196:1–196:48, 2024.

[8] E. P. Enoiu, D. Truscan, A. Sadovykh, and W. Mallouli, "VeriDevOps Software Methodology: Security Verification and Validation for DevOps Practices," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ARES '23, (New York, NY, USA), pp. 1–9, Association for Computing Machinery, Aug. 2023.

[9] M. Savary-Leblanc, L. Burgueño, J. Cabot, X. Le Pallec, and S. Gérard, "Software assistants in software engineering: A systematic mapping study," *Software: Practice and Experience*, vol. 53, no. 3, pp. 856–892, 2023.

[10] S. Santhanam, T. Hecking, A. Schreiber, and S. Wagner, "Bots in software engineering: a systematic mapping study," *PeerJ Comput Sci*, vol. 8, p. e866, Feb. 2022.

[11] R. Moguel-Sánchez, C. S. Martínez-Palacios, J. O. Ocharán-Hernández, X. Limón, and J. Sánchez-García, "Bots and their uses in software development: A systematic mapping study," in *2022 10th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pp. 140–149, 2022.

[12] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "Intellicode compose: code generation using transformer," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, (New York, NY, USA), p. 1433–1443, Association for Computing Machinery, 2020.

[13] A. Fontes and G. Gay, "The integration of machine learning into automated test generation: A systematic mapping study," *Software Testing, Verification and Reliability*, vol. 33, no. 4, p. e1845, 2023.

[14] M. Ciniselli, N. Cooper, L. Pascarella, A. Mastropaolo, E. Aghajani, D. Poshyvanyk, M. Di Penta, and G. Bavota, "An empirical study on the usage of transformer models for code completion," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, pp. 4818–4837, 2022.

[15] D. Drain, C. Wu, A. Svyatkovskiy, and N. Sundaresan, "Generating bug-fixes using pretrained transformers," in *Proceedings of the 5th ACM SIGPLAN International Symposium on Machine Programming*, MAPS 2021, (New York, NY, USA), p. 1–8, Association for Computing Machinery, 2021.

[16] B. Berabi, J. He, V. Raychev, and M. Vechev, "Tfix: Learning to fix coding errors with a text-to-text transformer," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 780–791, PMLR, 18–24 Jul 2021.

[17] Z. Li, S. Lu, D. Guo, N. Duan, S. Jannu, G. Jenks, D. Majumder, J. Green, A. Svyatkovskiy, S. Fu, and N. Sundaresan, "Automating code review activities by large-scale pre-training," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, (New York, NY, USA), p. 1035–1047, Association for Computing Machinery, 2022.

[18] P. Thongtanunam, C. Pornprasit, and C. Tantithamthavorn, "Autotransform: Automated code transformation to support modern code review process," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pp. 237–248, 2022.

[19] R. Tufano, S. Masiero, A. Mastropaolo, L. Pascarella, D. Poshyvanyk, and G. Bavota, "Using pre-trained models to boost code review automation," in *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, (New York, NY, USA), p. 2291–2302, Association for Computing Machinery, 2022.

[20] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. M. Zhang, "Large language models for software engineering: Survey and open problems," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, (Los Alamitos, CA, USA), pp. 31–53, IEEE Computer Society, may 2023.

[21] X. Huang, W. Liu, X. Chen, X. Wang, H. Wang, D. Lian, Y. Wang, R. Tang, and E. Chen, "Understanding the planning of llm agents: A survey," 2024.

[22] R. Kaur, D. Gabrijelčič, and T. Klobučar, "Artificial intelligence for cybersecurity: Literature review and future research directions," *Information Fusion*, vol. 97, p. 101804, 2023.

[23] N. Mahmud, C. Seceleanu, and O. Ljungkrantz, "Resa: An ontology-based requirement specification language tailored to automotive systems," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pp. 1–10, 2015.

[24] N. Mahmud, C. Seceleanu, and O. Ljungkrantz, "Resa tool: Structured requirements specification and sat-based consistency-checking," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1737–1746, 2016.

[25] N. Mahmud, C. Seceleanu, and O. Ljungkrantz, "Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic," in *Software Engineering and Formal Methods (SEFM)*, (Cham), pp. 332–348, Springer, 2017.

[26] M. E. Salari, E. P. Enoiu, W. Afzal, and C. Seceleanu, "An experiment in requirements engineering and testing using EARS notation for PLC systems," in *IEEE International Conference on Software Testing, Verification and Validation, ICST 2023 - Workshops, Dublin, Ireland, April 16-20, 2023*, pp. 10–17, IEEE, 2023.

[27] H. Bruneliere, V. Muttillo, R. Eramo, L. Berardinelli, A. Gómez, A. Bagnato, A. Sadovykh, and A. Cicchetti, “AIDOaRt: AI-augmented Automation for DevOps, a model-based framework for continuous development in Cyber–Physical Systems,” *Microprocessors and Microsystems*, vol. 94, p. 104672, Oct. 2022.