Graphical Abstract

SCENE: Guidelines for Security Chaos Engineering based on a Systematic Literature Review

Rodi Jolak , Mazen Mohamad , Ramana Reddy Avula , Jason Meek , Alexander Åström

Highlights

SCENE: Guidelines for Security Chaos Engineering based on a Systematic Literature Review

Rodi Jolak , Mazen Mohamad , Ramana Reddy Avula , Jason Meek , Alexander Åström

- SCENE is a comprehensive set of guidelines for systematically reporting security chaos engineering techniques.
- SCENE supports the clarity, consistency, and reproducibility of security chaos engineering practices.
- SCENE is considered as complementary to established reporting guidelines, closing the gap between research and practice.

SCENE: Guidelines for Security Chaos Engineering based on a Systematic Literature Review

Rodi Jolak [a,c,i], Mazen Mohamad [a,d], Ramana Reddy Avula [a], Jason Meek [b] and Alexander Åström [b]

[a] *RISE Research Institutes of Sweden , Gothenburg , Sweden*
[b] *Volvo Trucks Technology and Industrial Division , Gothenburg , Sweden*
[c] *Mid Sweden University , Östersund , Sweden*
[d] *Chalmers University of Technology , Gothenburg , Sweden*

ARTICLE INFO

Keywords:
Software Engineering
Security Chaos Engineering
Vulnerability Analysis
Resilience
Guidelines

ABSTRACT

Security Chaos Engineering (SCE) is a proactive approach to identify vulnerabilities and enhance security of systems. It embraces continuous security experimentation to build confidence in the capability of systems to withstand malicious conditions. Different SCE techniques are proposed for enhancing the resilience of software systems. The diversity of SCE techniques indicates the need for their collective analysis to uncover valuable practices and potential research opportunities. To fulfill this need, we consolidate and unify the knowledge on SCE practices through a systematic literature review. The results show that there has been limited and unsystematic investigation of SCE by the community, highlighting the importance of creating and promoting guidelines for SCE practices. Therefore, we create SCENE, a comprehensive set of guidelines for systematically reporting SCE. The goal is to support the clarity, consistency, and reproducibility of SCE practices. SCENE guidelines are evaluated by cybersecurity practitioners and active researchers in the field, and is mapped to established methodological guidelines. The results indicates that SCENE is perceived positive in terms of usefulness, understandability, practicality, and completeness. SCENE is also found to complement established experimental reporting guidelines and bridge the gap between academic studies and industrial use.

1. Introduction

Software-intensive Systems (SiS) are systems in which software significantly influences the design, deployment, and evolution of the system as a whole [36]. The underlying digital infrastructure and communication systems of SiS are highly susceptible to cybersecurity attacks due to their interconnected nature and reliance on various communication networks and third-party supply chain technologies [11]. Cybersecurity attacks pose a significant challenge to the digitalization of industry by threatening the confidentiality, integrity, and availability of critical systems and data, thereby hindering innovation, disrupting operations, and increasing the risk of economic losses. Regular cybersecurity assessments and continuous validation approaches are thus crucial to safeguard against the evolving cybersecurity threats to SiS.

Security testing demands specific expertise, making it challenging, often subjective, and less amenable to automation [37]. Generally, testing enables the evaluation of established capabilities and system attributes. On the other hand, experimentation provides means to uncovering unknown disturbances objectively, observing, and learning from failure [32]. Enterprises aiming to enhance the cybersecurity resilience of their systems and infrastructures are transiting from merely testing the security to embracing *continuous security experimentation* as part of the DevSecOps (Development, *Security*, and Operations) loop. This is important to proactively identify and mitigate potential vulnerabilities to real-world threats, ensuring system as well as infrastructure resilience.

One approach that embraces experimentation to enhance resiliency and identify weaknesses in SiS is *Chaos Engineering*. Chaos engineering is the discipline of experimenting on complex systems to build confidence in their capability to withstand disturbances [30]. It provides principles and practices to proactively uncover unknown failures before they manifest into business-impacting problems. Through experimentation, chaos engineering supports the identification of failures that are difficult, if not impossible, to find by traditional testing methods. These include failures caused by factors that developers do not understand (i.e., known unknowns) or factors that developers are neither aware of nor understand (i.e., unknown unknowns).

[i]Corresponding author
rodi.jolak@ri.se (.R.J.); mazen.mohamad@ri.se (.M.M.); ramana.reddy.avula@ri.se (.R.R.A.); jason.meek.2@volvo.com (.J.M.); alexander.astrom@consultant.volvo.com (.A.Å.)
www.rodijolak.com (.R.J.)
ORCID(s): 0000-0001-5656-9253 (.R.J.); 0000-0002-3446-1265 (.M.M.); 0000-0001-9672-2689 (.R.R.A.); 0000-0002-1442-0298 (.J.M.); 0009-0006-3879-4573 (.A.Å.)

Recent work has applied chaos engineering principles to support the design, build, and operation of complex systems that are more resilient to cyberattacks, marking a significant transition towards Security Chaos Engineering (SCE). According to [32] and [39], SCE is the discipline of instrumentation, identification, and remediation of failure within security controls through proactive experimentation to build confidence in the system's ability to defend against malicious conditions.

1.1. Problem and Contribution

Different SCE techniques are proposed in literature for enhancing the resilience of software systems. The diversity of SCE techniques indicates the need for their collective analysis to uncover valuable practices and potential research opportunities. To fulfil this, we consolidate and unify the knowledge on SCE practices by conducting a Systematic Literature Review (SLR). The aim is to (i) support researchers and practitioners identify key trends, challenges, and best SCE practices, and (ii) highlight gaps in current knowledge, guiding future research and development efforts to advance the field. In this paper, we address the following research questions:

- **RQ1.** *What techniques are available for supporting resilience using security chaos engineering (SCE)?*

  First, we review the literature to identify relevant SCE techniques with the goal to provide a comprehensive overview of these techniques. In particular, we review 359 studies and identify 19 SCE techniques. The results are reported in Section 3. Second, we ensure the reliability of the review by performing a quality control on 10.5% of the data (i.e., 30 studies).

- **RQ2.** *How can we categorize the techniques supporting resilience using security chaos engineering?*

  First, we analyse the identified techniques by the systematic literature review. Second, we categorize these techniques based on different aspects such as applicability, technical characteristics, effect, and evaluation. As a result, we develop the *Security Chaos Engineering guideliNEs* (SCENE) for systematically reporting SCE techniques. The aim of SCENE is support the clarity, consistency, and reproducibility of SCE practices.

- **RQ3.** *To what extent do the SCENE guidelines align with established methodological guidelines, and how are they perceived by researchers and practitioners in terms of usefulness, understandability, practicality, and completeness?*

  First, we review and map the SCENE guidelines to the widely used guidelines by Jedlitschka et al. [12] for reporting experiments in software engineering. Second, we conduct an evaluation of the SCENE guidelines involving a mix of twenty researchers and practitioners. We report the results of the evaluation in Section 6.

The key contributions of this paper include a comprehensive SLR of SCE studies and techniques and introduction of SCE-tailored guidelines that extend established methodological frameworks for software engineering experiments. The paper further advances the state of the art by explicitly addressing security-specific aspects of chaos engineering and the integration of AI and large language models that have not been systematically studied in prior work. The contribution also includes a publicly available, continuously updated repository that enables ongoing reporting of emerging SCE techniques.

The paper is structured as follows: Section 2 describes the background & related work. Section 3 details the approach. Section 4 presents the results of the SLR, and Section 5 presents the SCENE guidelines. The evaluation of SCENE is described in Section 6. Finally, Section 7 provides concluding remarks and discusses directions for future work.

2. Background and Related Work

In 2011, Netflix transitioned its services to the AWS cloud, which led to the development of the concept of chaos engineering [2]. Concerned about potential failures of internal instances during the migration, the engineers of Netflix created ChaosMonkey[1] to ensure system stability by introducing faults that randomly shut down internal instances. According to [2], chaos engineering entails conducting experiments on a distributed system to ensure it can handle disruptive conditions in a production environment. These conditions might include hardware failures, unexpected spikes in client requests, or incorrect values in runtime configuration parameters.

The Chaos Toolkit[2] is an open-source project that offers an extensible toolkit for conducting experiments. It allows developers to customize these experiments to suit specific use cases. The first security-specific tool to appear is ChaoSlingr[3]. It is a security chaos engineering tool focused primarily on the experimentation on cloud infrastructures to bring system security weaknesses to the forefront.

SCE aims to ensure that systems can gracefully adapt and recover from security incidents, identifying and mitigating vulnerabilities before they can be exploited, and fostering a culture of ongoing learning and adaptation to evolving threats [32]. Figure 1 shows the process of conducting SCE. It typically involves several

[1] ChaosMonkey: https://github.com/Netflix/chaosmonkey
[2] Chaos Toolkit: https://github.com/chaostoolkit
[3] ChaoSlingr: https://github.com/Optum/ChaoSlingr

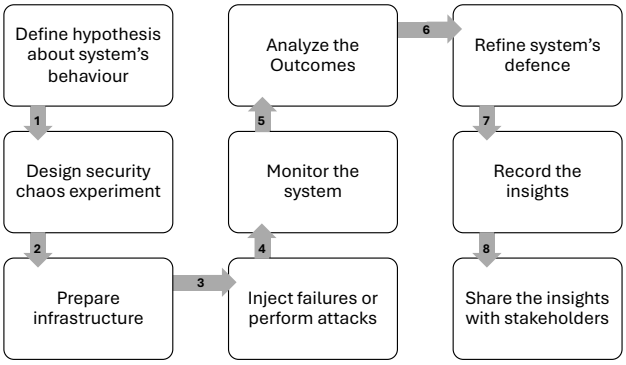

Figure 1: Typical process for conducting security chaos engineering.

key steps [39, 32]. First, hypotheses about the steady state of a system and how it should behave under various failure conditions are defined. After that, chaos experiments to test these hypotheses are designed and planned. The necessary infrastructure is then prepared to conduct the experiments. Next, controlled failures are introduced, and the system's response is monitored. The outcomes are analysed to identify weaknesses and areas for improvement. Based on the outcomes, system defences and security measures are refined. Lastly, the insights are recorded and shared them with relevant stakeholders to inform future strategies.

Recently, in [26], a comprehensive literature review of chaos engineering was conducted by analysing 88 works published between January 2019 and April 2024, encompassing both academic and grey literature sources. By synthesizing various definitions from these studies, the authors propose a unified definition of chaos engineering, emphasizing its role in resilience testing through the controlled injection of faults. It further explores the key functionalities and components of chaos engineering, outlines its fundamental principles and quality requirements, and proposes a taxonomy categorizing relevant tools and adoption practices. Notably, chaos engineering received increased attention between 2019 and 2024, with academic publications peaking in 2021, while contributions from grey literature steadily expanded throughout the period. Furthermore, an analysis of academic venues revealed that although many studies were published in unranked sources, a substantial proportion appeared in highly regarded Q1 venues, reflecting strong academic interest and recognition of the importance of this domain.

While [26] conducts a literature survey on chaos engineering with broader goals of understanding why and how chaos engineering is used, our work focuses on *security-specific* chaos engineering by explicitly targeting the techniques designed to enhance security resilience and provides guidelines for reporting such techniques. Indeed, no prior academic work has proposed comprehensive guidelines for conducting *security* chaos engineering, making our contribution the first of its kind in the literature.

Table 1 highlights the difference between our and related work. Existing literature provides essential foundations for understanding chaos engineering broadly, but none offer methodological guidance tailored to the security dimension. The multivocal review by Owotogbe et al. [26] offers a comprehensive synthesis of general chaos engineering, clarifying its definition, core activities, and components, while mapping challenges, trends, and tools across diverse contexts. However, its scope does not extend to security-focused techniques or to the formal structuring of security-specific experimentation practices. Moreover, practitioner-oriented sources such as Shortridge and Rinehart [32] provide conceptual overviews of SCE, but do not follow systematic review methods nor offer prescriptive reporting structures. In contrast, our work focuses exclusively on a detailed and systematic analysis of SCE-specific studies. Moreover, we develop SCENE, a first dedicated reporting guidelines for SCE. These guidelines aim to operationalize established principles for the unique characteristics of security-oriented chaos experimentation. Finally, unlike prior work, SCENE guidelines undergo validation by active cybersecurity practitioners and researchers. This validation enhances SCENE methodological robustness and real-world applicability.

3. Methodology

The methodology used by this study is shown in Figure 2. It starts by conducting a systematic literature review (SLR), following the guidelines of Kitchenham et al. [19]. The findings of the SLR are then used to develop guidelines for systematically reporting SCE. Finally, the resulting guidelines are validated by evaluating its usefulness, understandability, practicality, completeness, and correspondence to other established methodological guidelines.

3.1. Performing the SLR

We conducted the search in four major scientific research libraries using their search engines. These were: IEEE Xplore, ACM Digital Library, Web of Science, and Elsevier Scopus. We did not include other search engines, e.g., Google Scholar, as a preliminary search revealed that the results overlap with the ones from our selected search engines to a large extent. Moreover, to mitigate the risk of missing relevant studies that are not indexed in the included search engines, we conducted both forward and backward snowballing [40].

3.1.1. Search String

To create the search string, we used two groups of keywords. The first one captured the main area for the targeted literature, namely, security, cybersecurity and resilience. While the second group focused on the targeted technique, i.e., chaos engineering with

Table 1
Comparison between the related work and this work

Dimension	Owotogbe et al. [26]	Shortridge and Rinehart [32]	Our work (SCENE)
Focus & Scope	General Chaos Engineering (CE). It explores CE definition, core activities, components, and challenges. Moreover, it maps tools, trends, and research directions.	Practitioner-oriented overview of SCE concepts and practices	Exclusive focus on Security Chaos Engineering (SCE) techniques. Moreover, reporting SCE guidelines
Methodology	Multivocal literature review	Narrative practitioner synthesis	Systematic literature review and guideline construction as well as validation
Security-Specific Content	SCE not treated as a distinct domain	Conceptual: not systematically reviewed or methodologically structured	In-depth analysis of SCE techniques; methodological and reporting guidance for SCE
Main Contribution	Broad CE overview; taxonomy of CE practices and trends	Practitioner advice; introductory SCE concepts	SCENE: first dedicated reporting guidelines for SCE, mapped to established methodological guidelines
Validation	Not conducted	Not conducted	Formal evaluation with cybersecurity researchers and practitioners

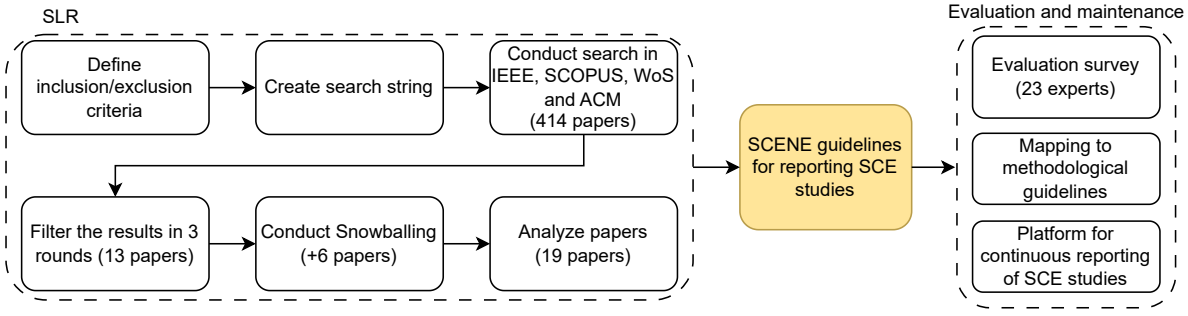

Figure 2: Depiction of the methodology followed to conduct this study.

the targeted activities, i.e., testing, fault injection and experimentation. The resulted search string looked as follows:

```
(Cybersecurity OR Security OR Resilience) AND (Chaos AND (Engineering OR Testing OR Fault OR Injection OR Experimentation))
```

3.1.2. Inclusion and Exclusion Criteria

The applied criteria are shown in Table 2. The first version of the criteria was created by three of the authors. The list was then refined and calibrated by applying it on a subset of the retrieved studies and engaging in discussion to clarify disagreements. This made applying the inclusion and exclusion criteria a straightforward task for the three involved authors.

For the exclusion criteria, we decided to disregard studies written in languages other than English (the shared working language among the authors) as well as vision and short studies as answering our research questions require studies that present substantive findings rather than preliminary ideas. Additionally, we restrict our included literature to studies published in the past 15 years, as chaos engineering was introduced in 2010, and earlier studies would naturally be irrelevant. Finally, we excluded studies that focus solely on fault injection in narrowly controlled scenarios, address chaos algorithms or cryptography, do not focus on security or cybersecurity, or are unrelated to software-intensive systems (SiS), as these topics are either not directly relevant or fall outside the defined scope of our review.

3.1.3. Result Filtering

To assess the consistency of applying the inclusion and exclusion criteria, an inter-rater reliability test was conducted. Three authors independently reviewed a random sample of 30 studies, representing approximately 10.5% of the total retrieved studies.

Table 2
Inclusion and exclusion criteria.

Inclusion criteria
1. Studies addressing the application of SCE.
2. Studies related to chaos fault injection.
3. Studies related to chaos testing.
4. Studies related to enhancing cybersecurity resilience through chaos engineering
Exclusion criteria
1. Studies written in any language other than English.
2. Studies published before 2010.
3. Short studies (less than 3 pages).
4. Vision / position studies.
5. Studies addressing chaos algorithms and cryptography.
6. Studies that are not focusing on security or cybersecurity.
7. Studies not related to software-intensive systems.
8. Studies focusing only on fault injection in specific controlled scenarios.

Table 3
Number of included studies after each round of filtration.

		After filtering round		
Library	**Studies**	**1st**	**2nd**	**3rd**
IEEE Xplore	73		7	4
ACM DL	65		5	1
Scopus	221		12	8
Web of Science	55		0	0
				+6 (snowballing)
Total	414	284	24	**19**

The analysis resulted in a Fleiss' Kappa value of 0.8, with $z = 7.59$ and $p = 3.24e * 14$, indicating "almost perfect agreement" according to Landis and Koch [20]. Following this initial analysis, the three authors held a discussion to resolve disagreements, clarify the criteria, and align their understanding of the review process. Once consensus was reached, the remaining 90% of the studies were independently screened by the same involved authors using the refined criteria.

The filtration was done on the 414 retrieved studies over three phases as shown in Table 3. In the first phase, duplicated studies were removed. The second phase was done by applying the inclusion and exclusion criteria (shown in Table 2) on the metadata of the studies, i.e., title, abstract, and keywords, resulting in a total of 24 studies. In the final round, the filtration was done based on the whole content of the studies, which resulted in a total of 13 studies identified as relevant. Subsequently, backward and forward snowballing techniques were applied, resulting in an additional 6 studies, summing up to a total of 19 studies for further analysis.

3.2. Data Extraction and Development of SCE Guidelines

The aim of this step is to develop guidelines for systematically reporting SCE techniques. After collecting the relevant studies, two authors discussed different aspects that can be used for categorizing the SCE techniques. The aim of this discussion was to prepare and plan for the process of data extraction. The discussed aspects are the following:

- What is the purpose of the SCE techniques, and what are the expectations?
- What are the experimentation strategies adopted by the SCE technique? What are the required inputs and resources?
- Which domains are the SCE techniques applicable to?
- How are the SCE techniques evaluated, and what are the evaluation results?

To systematically extract relevant details for the development of the SCE guidelines, we read the relevant studies identified by the literature review and snowballing search. By doing so, different details and aspects have emerged and been noted. These details are then organized in themes by conducting a thematic analysis at the explicit level [6]. This allowed us to identify recurring and stable reporting practices across heterogeneous SCE studies, as well as systematically observe aspects that are inconsistently reported or absent in the existing literature.

By jointly considering reported details and recurring omissions, the analysis yielded cross-cutting themes that capture foundational aspects of SCE techniques, rather than context- or domain-specific characteristics tied to individual studies. These themes were subsequently consolidated into a set of high-level categories and sub-categories that structure the proposed reporting guidelines. These resulting categories are shown in Table 4, which also provides examples and/or descriptions of the categories.

The applicability category helps to gain a comprehensive overview of the SCE techniques. We survey the motivation and goal of using the techniques along with the application domain The other categories give an insight into the technical details of the SCE techniques and the effect they have on cybersecurity in terms of targeted threats, types of inspected faults and additional relevant insights or lessons learned from their application. Finally, we survey the evidence provided regarding the validity of the techniques. Specifically, we survey the method of evaluation, the used metrics and the reported results. These characteristics allow us to systematically categorize the techniques based on different aspects, e.g., clarity, consistency, and reproducibility and propose guidelines for reporting them.

Table 4
Categorization of the techniques for supporting resilience using security chaos engineering

Category	Sub-Category	Description or Example
Applicability	Identified Security Challenges	The motivation for using SCE for cybersecurity, e.g., limitations of other approaches and increasing confidence in system resilience.
	Contribution/Goal	The main objective of the study and the main contribution, e.g., a SCE framework for early threat detection, improving of resilience, or identification of vulnerabilities.
	Application Domain	E.g., general purpose, IoT, cloud.
Technical Characteristics	Prerequisites	E.g., system design, specific data.
	Required Resources	Required hardware and software resources, e.g., sensors, virtual machines.
	Attack Scenarios	Methodology used to derive attack scenarios, e.g., attack trees, attack goals.
	SCE Approach	A description of the SCE approach including main steps and processes.
	TRL	The Technical Readiness Level [22] of the proposed approach.
	Use of AI	Whether the approach utilizes artificial intelligence.
Effect	Targeted Security Threats/Attacks	What STRIDE threats are targeted.
	Type of Injected Faults	E.g., miss configuration, data/resource injections.
Evaluation	Method	Analytical[a], Empirical[b], Both.
	Description	Description of how the evaluation was conducted.
	Metrics	Quantitative and qualitative criteria and metrics used in the evaluation e.g., time to complete attack, security performance.
	Results of the Evaluation	The results of the evaluation in terms of the evaluation criteria.
	Lessons Learned	Pros, cons, side effects.

[a]An analytical evaluation focuses on theoretical analysis or formal methods.
[b]An empirical evaluation involves experimentation or observation, often using real-world deployments, simulations, or prototypes.

3.3. Evaluation of the SCE Guidelines

The SCENE guidelines are evaluated in two ways. First, we validated the guidelines with cybersecurity researchers and practitioners. Second, we mapped the SCENE guidelines to established methodological guidelines in the field.

3.3.1. Feedback from Researchers and Practitioners

We apply a mixed-methods approach that integrates both quantitative and qualitative techniques for data collection and analysis, as outlined by [7]. This methodology allows us to explore strengths and limitations of the guidelines, including the social and cognitive aspects associated with its use. Our data collection strategy involves a semi-structured questionnaire (see Table 5) including questions on the usefulness, understandability, practicality, and completeness of the SCENE guidelines.

The target groups for feedback collection comprises practitioners and researchers with expertise and/or experience in cybersecurity and secure software/system engineering. For this study, the accessible population is a subset of the targeted group identified opportunistically through convenience sampling (i.e., via networks of collaborators and contacts). Additionally, the authors of the papers identified by the SLR are also considered as accessible and relevant target group for the evaluation of SCENE guidelines. The accessible population that is invited to evaluate the SCENE guidelines has between 5 and 20 years of professional experience in cybersecurity.

To preserve participant anonymity and minimize potential biases, we deliberately chose not to collect background information such as demographic or professional details. This decision enhances construct validity and helps mitigating threats to internal validity, such as social desirability bias or stereotype effects, which can influence how participants respond. Moreover, it strengthens ethical considerations around privacy and data protection, and respect for anonymity and confidentiality [33].

Prior to responding to the evaluation questions, the participants were provided with a brief overview of the SCE methodology and the SCENE guidelines to ensure a shared understanding of the context[4]. Out of 75 invited individuals, a total of 23 (i.e., response rate of

[4]Evaluation Questionnaire: https://forms.gle/b9N1nmtgr7WhRQ6d8

Table 5
Questions used to collect feedback on the SCENE guidelines from cybersecurity researchers and practitioners.

Question Text	Question Type
Q1. Usefulness: the guidelines help creators of security chaos engineering (SCE) techniques in promoting a clearer reporting, better reproducibility, and improved communication of the developed technique.	Closed (1 Strongly Disagree to 5 Strongly Agree)
Q2. Any comments related to the Usefulness of SCENE?	Open
Q3. Understandability: the guidelines can be easily interpreted and applied by creators and users of SCE techniques.	Closed (1 Strongly Disagree to 5 Strongly Agree)
Q4. Any comments related to the Understandability of SCENE?	Open
Q5. Practicality: the guidelines can be realistically followed given typical constraints e.g., data availability or organizational policies.	Closed (1 Strongly Disagree to 5 Strongly Agree)
Q6. Any comments related to the Practicality of SCENE?	Open
Q7. Completeness: the guidelines cover all the essential elements needed to comprehensively report a security chaos engineering technique.	Closed (1 Strongly Disagree to 5 Strongly Agree)
Q8. Any comments related to the Completeness of SCENE?	Open
Q9. Final Comments: Any final comments that you would like to share with us.	Open

30.6%) provided feedback on the SCENE guidelines. To analyse the collected feedback, we employ a sequential explanatory approach [7], where qualitative findings are used to enrich and clarify the interpretation of the quantitative results.

3.3.2. Mapping of SCENE to Established Guidelines

The SCENE guidelines proposed in this study aim to identify important characteristics for reporting SCE studies. However, since SCE is a technique rather than a research method, it is important to ensure that the SCENE guideline does not contradict existing and established methodological guidelines for conducting studies and reporting them in software engineering. Rather SCENE should be compatible and complementary to those guidelines. Since in most cases, SCE is conducted and reported as an experiment. we review and map the SCENE steps to the widely used guidelines by Jedlitschka et al. [12] for reporting experiments in software engineering.

The mapping was first conducted by one author, who reviewed each part of SCENE and suggested where it would be reported according to Jedlitschka et al.'s guidelines. A session with three authors was then held to review and refine the mapping through discussion and consensus.

To reduce the risk of bias from relying on a single source, we further considered the review of these guidelines by Kitchenham et al. [18]. Their evaluation identified several questions not addressed by Jedlitschka et al.'s guidelines, particularly regarding the relevance of results to practitioners and their applicability in industrial contexts. We therefore examined how SCENE could contribute to addressing these gaps. We applied the same process here: one author drafted an initial mapping of how SCENE could address these missing aspects, which was subsequently discussed and adjusted in a session involving three authors.

4. SLR Results and Discussion

The 19 surveyed studies meeting the search, inclusion, and exclusion criteria discussed in the previous section are listed in Table 6, which includes main characteristics of these studies such as their application domain, Technology Readiness Level (TRL) [22], usage of AI, targeted threats, use of attack scenarios, type of fault injections, and evaluation method. In the following, we describe the data extracted from these studies according to the categories listed in Table 4.

C1. Applicability - Identified Challenges: Four main themes can be seen throughout the studies in relation to identified challenges. These are ***lack of existing solutions***, ***human error and subjectivity***, ***monitoring and securing SiS***, and ***production system uptime***. The lack of existing solutions theme includes the lack of security, resilience, and auditing techniques, as well as the lack of standardization and methodologies around SiS testing, especially in relation to SCE. Eight studies list this theme as a challenge. E.g., [8] and [10] fall within this theme. The second theme, human error and subjectivity, covers errors, bias, and subjectivity around human-made security analyses and systems development. It is listed in five studies as a challenge, including [3] and [27]. The third theme is monitoring and securing SiS. This theme includes the challenges associated with securing, monitoring, maintaining, and trusting of modern complex systems and is seen in nine of the studies. The final theme, production system uptime, is listed in only two studies, but in each is a core theme [5] and [28]. These studies emphasize the challenges that even small amounts of downtime bring, and the financial losses associated with this.

Table 6
Studies included in this review, along with their main characteristics.

Study	Year	Domain	TRL [22]	AI	Targeted Threats [a]	Attack Scenarios	Fault Injection [b]	Evaluation Method
[39]	2020	Cloud	4-6	No	S T R I D E	Yes	T1, T2	Empirical
[3]	2023	Cloud	4-6	Yes	E	Yes	T2, T3	Empirical
[5]	2023	General Purpose	4-6	No	S T R I D E	Yes		Empirical
[34]	2022	Cloud	*NA*[c]	No	D	No	T1	*NA*
[4]	2024	General Purpose	4-6	Yes	S T R I D E	Yes	T2, T3, T5	*NA*
[8]	2023	Cyber-Physical Systems	4-6	No	T D	No	T1, T2, T4, T5, T6	Empirical
[27]	2023	General Purpose	4-6	No	T	Yes		Empirical
[1]	2022	Systems of Systems	1-3	No	T D	Yes	T1, T4	Empirical
[25]	2022	General Purpose	1-3	No	T I D	Yes	T4, T5	Empirical
[10]	2024	Operational Technologies	4-6	No	S T I D E	No	T1, T2, T4, T5	Empirical
[28]	2021	Open Systems Architecture	1-3	No	D	No	T5, T6	Empirical
[35]	2021	Cloud	1-3	No	D	No	T5	*NA*
[41]	2021	Internet of Things	4-6	No	I D	No	T5	Empirical
[14]	2018	Cloud	4-6	No	D	No	T4	Analytical+Empirical
[38]	2021	Cloud	4-6	No	T I	No	T1, T5	Empirical
[31]	2024	Cloud	4-6	No	D	No	T1	Empirical
[15]	2024	Internet of Things	1-3	No	D	No	T5	Empirical
[21]	2019	Cloud	*NA*	No		No		*NA*
[9]	2023	Cloud	4-6	No	D	No	T1, T4, T5	Empirical

[a] **STRIDE**: **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege.
[b] **T1**: Disable functionality or system elements; **T2**: Manipulation of configuration; **T3**: Credentials misuse; **T4**: Various Denial of Service (DoS) injections; **T5**: Data and resource injections; **T6**: Environmental related.
[c] ***NA***: Not Available.

C2. Applicability - Contribution/Goal: Six main themes can be observed. These are ***framework/ methodology creation***, ***testing of security systems***, ***system resilience assessment***, ***vulnerability identification***, ***resilience improvement***, and ***informational***, The most common theme is framework/methodology creation. This theme encompasses cases where creation of SCE based frameworks/methodologies/testbeds/tools is the primary research contribution. Twelve studies list contributions under this theme, with [3], [10], and [41] being such examples. The assessment of system resilience is characterized by using SCE tools/frameworks to assess system or item resilience. Nine studies follow this theme. Next is vulnerability identification. This includes cases where SCE tools/frameworks are used to identify vulnerabilities in systems. This theme is present in six studies. The resilience improvement theme is observed in five studies. It includes the use of SCE tools to assess other tools that are in place to improve resilience, [14] and [31], as well as directly using SCE frameworks to improve system resilience, [4], [8], and [27]. Testing of security tools, is characterized by using existing SCE tools/frameworks to test the effectiveness of various security systems. Four studies follow this theme, three of which focus strongly on the creation of IDS/monitoring systems that SCE is then used to verify/validate, [14], [38], and [31]. The last theme is informational. Two studies [34] and [21] contribute mainly to information sharing in relation to SCE.

C3. Applicability - Application Domain: Several application domains have been observed in the research studies and can be grouped into the following themes. ***Cloud***, ***IoT***, ***General***, and ***Other***. The cloud domain includes cloud infrastructure, cloud platforms, and cloud-native applications. General covers cases where the application of the approach is not limited to a particular domain. It, for example, could include the domains listed here as well as other unlisted domains. The IoT domain covers approaches applied to IoT devices/systems. The final domain, "other", includes individual application domains that did not fit any of the other domains. The distribution of domains between studies can be seen in Table 6.

C4. Technical Characteristics - Prerequisites: The two most distinguished themes related to prerequisites for the methods or tools in scope in the studies can be described as ***Design documents*** and ***Dynamic run-time data***. In the design documents theme we include user stories, functional requirements, network diagrams / topology specification, system context, models and configurations. There are five studies that mention these as prerequisites, [3], [5], [4], [25] and [35]. The second theme, dynamic run-time data, is comprised of sensor data which are used during the executions. It is part of three studies, [8], [25] and [15].

C5. Technical Characteristics - Required Resources: This category can be described using three themes ***Software applications***, ***Hardware*** and ***Infrastructure***. The software applications theme includes all additional software applications that are needed in the study's proposed tool/method, but excluding the tool itself. One such example is the usage of an additional tool for threat modelling, see [3]. Twelve studies fit this theme. Seven studies are relevant for the hardware theme, which basically states if the study specifies any specific requirement on hardware resources. Some studies require specific hardware resources for the

test environment, see [1] and [25]. The third theme, infrastructure, describes if the method proposed in a paper utilizes any infrastructure solutions, such as Cloud solutions [39], [3], [14], [38] and [9]. In one occasion an on-line large language model is utilized [4].

C6. Technical Characteristics - Attack Scenarios: This category consists of one theme which is equivalent to the category itself, ***Attack scenarios***. It includes different ways in which attacks can be learned from previous attacks or new attack scenarios need to be derived as part of the methodology. This could include conducting an attack tree, defining attack goals, or describing failure scenarios. Table 6 presents the studies in which this is explicitly mentioned.

C7. Technical Characteristics - SCE Approach: Most of the studies have their specific touch on the approach; however, there are common themes: ***Fault injection***, ***Hypothesis***, ***Steady state*** and ***Validation of security controls***. Fault injection, i.e., introducing faults into the system to see how it responds, is a common shared theme, it is also a specific category, see ***C12*** below. Two themes are related, hypothesis and steady state, each mentioned in [39] and [1]. The former is about making hypotheses about how the system works, also mentioned in [8], while the latter is about defining the steady state and is additionally mentioned in [28]. The theme validation of security controls, is part of two studies, [3] and [4].

C8. Technical Characteristics - TRL: The Technology readiness levels (TRL)[22], which is a methodology for estimating technology maturity, were assessed for all studies, with the vast majority belonging to TRL 4 - 6. The remaining ones are either in TRLs 1-3 or were not assessed. The results can be seen in Table 6.

C9. Technical Characteristics - Use of AI: Of all the studies, 17 did not make use of AI in the approach, while 2 did. *"Using an NLP model trained on user stories to identify attack goals"*, and *"constructing a threat model based on an attack-defence tree"* are the two observed uses of AI. Table 6 shows which studies made use of AI or not.

C10. Effect - Targeted Security Attacks or Threats: The distribution of targeted security attacks or threats, in terms of STRIDE categories, is shown in the *Targeted Threats* column of Table 6, where S is Spoofing, T is Tampering, R is Repudiation, I is Information Disclosure, D is Denial of Service, and E is Elevation of Privilege. D (15) is the most commonly mentioned threat and is associated with system availability. T (8), I (7), and E (7) are the second most mentioned threats and have similar occurrence. The least mentioned threats are S (5) and R (3).

C11. Effect - Type of Injected Faults: Types of injections can be described using six themes. ***T1 - Disable functionality or system elements*** includes different kinds of terminating functionality such as processes or system elements, these are used in 9 studies. The second one is ***T2 - Manipulation of configuration***, which includes changes in security control configuration or other configurations, part of five studies. ***T3 - Credentials misuse***, includes misusage of credentials or escalation of privileges, and is part of two studies. ***T4 - Various DoS injections***, this theme includes aspects such as overloading systems, flooding networks, delaying packets, and various system disruptions, among others. This is considered in seven studies. The ***T5 - Data and resource injections*** theme is a compilation of various fault injections, including malicious code, data/SW/HW faults, java-faults, and API-faults, among other things. It is included in eleven studies. The last theme ***T6 - Environmental related***, used in two studies, is constituted by fault injections relating to the environment of the system. The mapping of fault injection type used in the studies can be seen in Table 6.

C12. Evaluation - Evaluation Method: When it comes to evaluating the suggested approach, 16 out of the 19 included studies provided some form of evaluation. Notably, all evaluations were empirical and included controlled experiments such as fault injection to observe system behaviour under stress and case studies, while only one study combined that with an analytical evaluation of risks [14]. Another observation is that none of the evaluations involved human experts as part of the assessment process.

C13. Evaluation - Evaluation description: Across the reviewed studies, several common themes emerged in how systems and approaches were evaluated. The evaluations were typically performed in **cloud-based or distributed environments**, e.g., [39, 27, 5], with Amazon Web Services (AWS) and Google Cloud Platform (GCP) frequently serving as testbeds. Other studies used **custom-built testbeds** populated with user stories, microservice setups, or industrial control scenarios, e.g., [8, 41, 15]. The evaluation focused on assessing **resilience, recovery, and performance under stress** [39, 38, 14, 3, 25, 31, 9] and targeting specific **security aspects** such as access control configurations, credential misuse, and vulnerability detection in cloud pipelines, e.g., [38, 27, 4]. In terms of data used for evaluation, most studies relied on **synthetically generated** workloads, simulated attack scenarios, or testbed-driven telemetry to assess system behaviour, e.g., [39, 3, 1, 31]. In some cases security artifacts, such as misconfigured access control [27] and code vulnerabilities [4] were used.

C14. Evaluation - Metrics: The majority of the included studies used **quantitative metrics** in their evaluations. **Time-based metrics** such as attack completion time, system response time, and recovery duration were used to assess operational efficiency [39, 38, 14, 31, 9]. **Resource utilization** was another common focus, with several studies measuring CPU and memory consumption to understand the overhead of fault

injection or recovery processes [39, 1, 9]. **Detection-related performance**, especially in terms of detection rate, false positives, false negatives, Mean Time to Detect (MTTD), and Mean Time to Recover (MTTR), was also used [5], as well as **fault detection and tolerance** metrics [28, 41]. Some studies applied **domain-specific** indicators, e.g., in industrial gas purification and IoT sensors [8, 15]. Other used metrics include credential misuse [3], packet loss between digital twins and physical systems [10] and risk analysis evaluation using Common Vulnerability Scoring System (CVSS) severity scores. **Qualitative metrics** were in a few studies used to evaluate access control and system integrity [27, 25], and overall security performance [39].

C15. Evaluation - Results of the Evaluation:

A common theme in the results is the assessment of the security and resilience of the systems under study. The SCE approach helped researchers to **demonstrate the resilience of their systems to threats** [3, 5, 38], and to **compare the security of different systems**, such as cloud providers [39]. Other studies reported a **demonstration of the impact** of the performed SCE, including assessments of resource consumption and performance [1, 9, 10], as well as the effect on system behaviour and output [15]. Some evaluations also **uncovered specific system vulnerabilities**, such as identifying components that failed to recover from injected attacks [8], or tracking encryption key leakage and inconsistent binary outputs under fault conditions [41].

C16. Evaluation - Lesson Learned: The majority of the reviewed studies conclude based on their evaluations that SCE is effective for assessing system resilience, identifying vulnerabilities, and enhancing overall security posture. However, some challenges and limitations remain, including the selection of performance metrics [1] and the cascading effect of chaos experiments due to potential lack of detailed understanding and knowledge about the targeted systems, [25]. Studies that use AI techniques, i.e., Natural Language Processing (NLP) and Large Language Models (LLMs) [3, 4], find that it is valuable to integrate AI for threat modelling and decision support during SCE experimentation. However, the study that uses LLMs highlights risks related to hallucination and nondeterministic nature of these models which might lead to incorrect output. Hence, the study emphasize the need for human oversight and expertise in order to provide proper context and verify LLMs' output. Generally, for approaches that use AI, a challenge is to maintain the models and to keep the data used for training updated, diverse and unbiased.

4.1. Statistics

A quantitative analysis was done based on the 19 primary studies that satisfied all inclusion criteria. The six most salient ordinal attributes (TRL, Domain, Targeted Threats, Fault-Injection types, AI usage, and Evaluation Method) extracted from these studies are listed in Table 7 and their yearly frequencies are plotted in Fig. 3-Fig. 8.

Table 7
Descriptive statistics of ordinal attributes across 19 studies.

Attribute / Characteristic	Count (%)
TRL level	
1–3	5 (26.3%)
4–6	12 (63.2%)
Not reported	2 (10.5%)
Domain	
Cloud	9 (47.4%)
General-Purpose	4 (21.1%)
Cyber-Physical Systems	1 (5.3%)
Systems of Systems	1 (5.3%)
Operational Technologies	1 (5.3%)
Open Systems Architecture	1 (5.3%)
Internet of Things	2 (10.5%)
Targeted threats	
S (Spoofing)	18 (94.7%)
T (Tampering)	18 (94.7%)
R (Repudiation)	17 (89.5%)
I (Information disclosure)	17 (89.5%)
D (Denial of service)	17 (89.5%)
E (Elevation of privilege)	5 (26.3%)
Fault-Injection types	
T1 (Disable functionality / system elements)	8 (42.1%)
T2 (Manipulation of configuration)	5 (26.3%)
T3 (Credentials misuse)	2 (10.5%)
T4 (DoS injections)	6 (31.6%)
T5 (Data / resource injections)	10 (52.6%)
T6 (Environment-related)	2 (10.5%)
Not reported	3 (15.8%)
AI usage	
Yes	2 (10.5%)
No	17 (89.5%)
Evaluation method	
Empirical	14 (73.7%)
Analytical	1 (5.3%)
Not reported	4 (21.0%)

The set of studies shows a clear emphasis on mid-to-high TRL systems, with 12 of the 19 studies (63.2 %) targeting TRL 4–6 prototypes, indicating that most research focuses on mature or near-deployment systems. Early-stage systems (TRL 1–3) are represented in five studies (26.3 %), all published between 2022 and 2024, highlighting a recent interest in applying SCE concepts to emerging and less mature platforms. In terms of domain coverage, Cloud platforms are the most frequently studied (47.4 %), followed by General-Purpose platforms (21.1 %). The studies largely cover the STRIDE elements, but Elevation of Privilege (E) is notably under-represented, appearing in only 5 of 19 studies (26.3 %).

Fault-injection types are observed to be clustered around certain types (T1, T5), while complex types

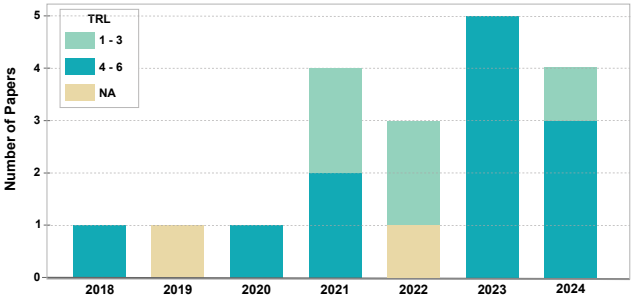

Figure 3: Yearly distribution of TRL levels of SCE techniques.

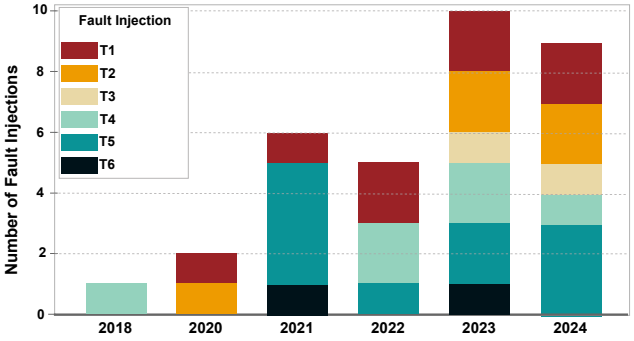

Figure 6: Yearly distribution of fault-injection types across studies.

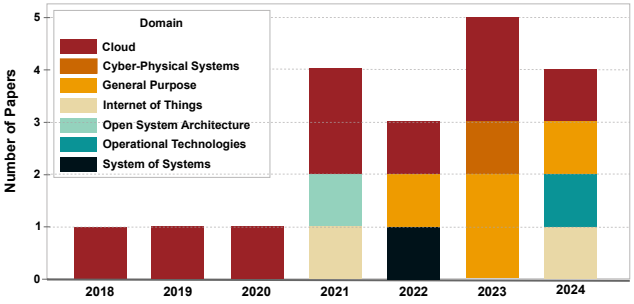

Figure 4: Yearly distribution of study domains.

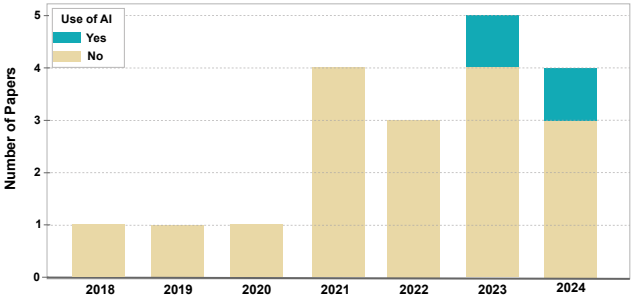

Figure 7: Yearly distribution of studies using AI-based SCE.

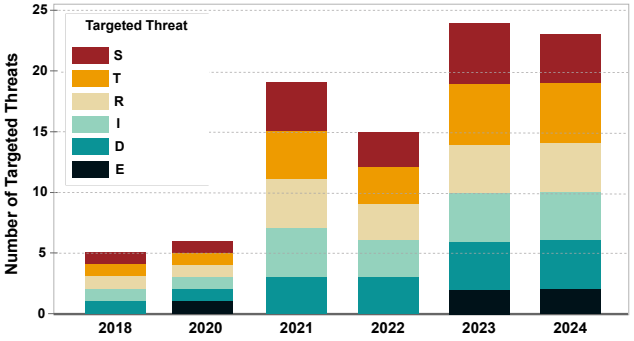

Figure 5: Yearly distribution of targeted threats (STRIDE) reported in the studies.

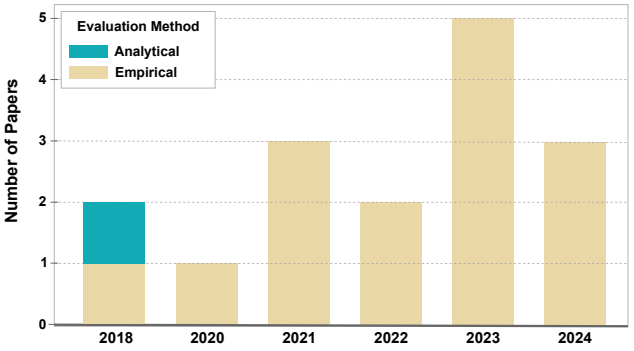

Figure 8: Yearly distribution of evaluation methods across studies.

such as T3 (Credentials misuse) and T6 (Environment-related faults) are much less common. This pattern reflects the use of simpler and widely supported fault templates over more complex environment-level injections. AI-assisted SCE remains rare, with only two studies (10.5 %) explicitly reporting AI support. Evaluation methods are predominantly empirical (73.7 %), while only one study (5.3 %) employs analytical techniques.

4.2. SLR Synthesis and Gaps

The studied research shows that there is a shortage of current SCE solutions for assessing the security and resilience of SiS. Additionally, the complexity of these systems makes them challenging to monitor, maintain, and secure. This is particularly pertinent for organizations operating large-scale production environments, which rely heavily on system uptime and have limited tolerance for disruption. As a result, SCE remains largely exploratory in such contexts, with *most reported evaluations confined to controlled or laboratory environments*. This indicates that there is an opportunity for SCE to enhance resilience and security while simplifying and automating assessment tasks. Furthermore, challenges in system development caused by human error and subjectivity present opportunities for the application of SCE.

The primary area of research and application is within the cloud domain, which can be expected since chaos engineering was pioneered in the realm of cloud services (e.g., Netflix). Cloud environments provide

Table 8
Summary of Synthesis and Research Gaps from the SLR

Observed Pattern	Explanation	Research Gaps
Dominance of cloud-based SCE	Mature automation, observability, isolation, and relatively low experimentation cost in cloud platforms enable safer and repeatable chaos experiments.	Methods tailored for other domains and tightly coupled, less observable systems (safety-/hardware-/regulation-constrained).
Limited adoption of AI-based SCE	Early tooling maturity, trustworthiness, and reproducibility, hallucination concerns, and a lack of high-quality domain data.	Robust, validated AI/LLM-supported SCE methods with transparency, benchmarking, and domain datasets.
Emphasis on DoS	DoS-like conditions are straightforward to trigger, observe, and quantify in experiments.	Under-representation of complex threats (e.g., Elevation of Privilege) that are harder to induce and measure via chaos.
Preference for fault injection over structured security modelling	Operational experiments are easier to run than building/maintaining comprehensive models (e.g., attack trees).	Integration of systematic modelling with SCE practices, leading to a movement toward standardized processes and metrics.
Limited real-world validation and expert involvement	Risk of disruption, restricted access to production data, and organizational constraints.	Field studies with practitioner participation, external validity evidence, and scalable, industrial-grade evaluations.
Sparse replication packages	Resource constraints and the early maturity of the field.	Reproducibility via artifacts, datasets, and open benchmarks to support cumulative evidence-building.

mature automation, observability, and isolation mechanisms, which lower the barrier for conducting chaos experiments and explain their dominance in the reviewed literature. In contrast, there is comparatively little research applying SCE to domains such as IoT or operational technologies, where experimentation is more constrained due to safety, hardware, or regulatory considerations. This suggests *a gap in adapting SCE techniques to less observable and more tightly coupled system environments*.

The studies highlight system resilience assessment and vulnerability discovery as the two main use cases of SCE, which align well with the identified challenges. Although resilience improvement is listed as a third use case, the two aforementioned use cases can also be expected to contribute indirectly. Resilience is closely related to availability (DoS), which is by far the most frequently included threat class, appearing in fifteen studies. In contrast, *Elevation of Privilege is significantly under-represented*, likely due to the difficulty of triggering and observing internal privilege transitions through chaos-style experiments.

Two security-related topics show an interesting contrast in the reviewed studies: the inclusion of traditional security methodologies and fault-injection techniques. The former, such as attack trees, is explicitly mentioned in only seven studies, while the latter appears in sixteen. This contrast suggests that *SCE research currently favours operational experimentation over structured security modelling*, potentially limiting systematic reasoning about attack paths and assumptions. At the same time, the wide range of applied methods indicates a lack of standardization in SCE practices.

An initiative by Bedoya et al. [3, 4] indicates a promising shift in SCE through the integration of AI and LLMs, enabling more automated and adaptable approaches. LLMs, in particular, show potential for automating tasks such as attack–defence tree generation and vulnerability identification, enhancing efficiency in DevSecOps workflows. However, further research is needed to address limitations of LLMs (e.g., hallucination) and to assess the applicability of LLM-supported SCE approaches in complex and domain-specific systems. *The rarity of AI-based SCE in the reviewed studies suggests that these approaches remain exploratory*, likely due to concerns related to trustworthiness, reproducibility, and the availability of high-quality training data.

A key limitation observed across the reviewed studies is the lack of human and domain expert involvement in validating SCE approaches. In addition, evaluations rarely employ real-world production data. This indicates that *many proposed techniques have yet to demonstrate external validity and scalability in operational settings*. Finally, the frequent absence of replication packages hinders independent validation and cumulative knowledge building, highlighting a *broader quality and maturity gap in current SCE research*.

A consolidated overview of these observations, their plausible explanations and the corresponding research gaps is summarized in Table 8. This synthesis highlights how practical constraints, methodological limitations,

and domain-specific factors shape the current landscape of SCE research. It also makes explicit the areas where future work is most needed to advance the maturity and applicability of SCE techniques.

4.3. Online Artifact - Live SLR Repository

The SLR presented in this paper is also made publicly available and continuously maintained in a GitHub repository [13]. The repository contains an up-to-date Excel spreadsheet (slr.xlsx) that holds all the extraction tables and metadata that were produced in the review. To keep the review *living*, we leverage GitHub's issue and action workflow. A custom issue template has been defined, allowing anyone to submit a new entry by providing structured data that corresponds to the columns in the Excel spreadsheet. A custom-defined GitHub Action automatically converts the issue payload into a new row in the spreadsheet in a new branch and creates a Pull Request. Repository maintainers review the PR, check for consistency, and, upon approval, merge it. This workflow makes it trivial for the community to contribute and audit, turning the SLR into a living artifact that evolves with the state of the field.

5. SCENE Guidelines

We developed the SCENE guidelines for systematically reporting SCE techniques and practices, based on the characteristics listed in Table 4 and shown in Figure 9. The guidelines synthesize insights from the examined literature and the identified reporting gaps, and aim to support SCE practitioners and researchers in systematically documenting, communicating, and assessing SCE techniques in a clear, consistent, and reproducible manner.

The SCENE guidelines are intentionally formulated at a higher level of abstraction. This reflects the current state of evidence in the SCE literature, in which the 19 identified studies are reported across heterogeneous application domains and levels of maturity, often with preliminary or context-specific evaluations and varying degrees of reporting completeness. Defining highly granular, domain- or maturity-specific reporting requirements at this stage would therefore risk overfitting the guidelines to limited and uneven empirical evidence. Accordingly, SCENE is designed as a foundational reporting guideline that remains applicable across different SCE maturity levels and application domains, while providing a stable structure that can be incrementally refined as the body of empirical evidence grows and reporting practices mature.

The reporting guidelines comprise four main categories: applicability, technical characteristics, effect, and evaluation, where are described along with their subcategories in detail in the following.

5.1. Applicability

The applicability category provides a comprehensive understanding of the technique's relevance, ensuring that stakeholders can assess the technique's value and suitability for their specific needs. This category outlines the following characteristics.

Identified Security Challenges: This characteristic is essential for establishing the relevance of the technique by clearly articulating the specific cybersecurity problems it addresses. Creators of SCE techniques should describe the challenge and its nature, affected assets, and the eventual impact. A challenge might be lack of measures for targeting of insider threats in health systems or lack of system architectural tactics for controlling privilege escalation in cloud infrastructures. Considering the nature of the challenge, this should be specified whether it involves technical vulnerabilities (e.g., insecure APIs), systemic issues (e.g., lack of visibility in distributed systems), or operational challenges (e.g., access control misconfigurations). Reporting should also include a specification and description of the affected assets (e.g., connectivity system, autonomous driving system) and the consequences or impact of not addressing the challenge (e.g., unauthorized users may gain access to sensitive data or services).

Reporting Identified Security Challenges

> The technique targets [*specific challenges]* and their [*nature*] related to a specific [*asset system*], which if not addressed lead to [*consequences*].

Contribution or Goal of the SCE Technique: The contribution or goal of an SCE technique defines its intended purpose and the value it brings to cybersecurity practice. Contributions can vary depending on the nature of the technique. Examples of contribution include; improving threat detection accuracy, reducing response time, enhancing system resilience, automating security tasks, or supporting compliance with specific standards. Reporters should report the contribution by describing both the *functional* and *strategic* goals of applying the SCE technique on the targeted system (i.e., use case). Functional goals are specific technical capabilities the SCE technique is designed to perform such as, detecting anomalous behaviour in real-time, encrypting data, or managing access controls. While strategic goals represent a broader impact or value the technique aims to achieve within an organizational or operational context, such as improving system resilience, reducing operational costs, or supporting regulatory compliance. It is important to provide a contextual information about the use case and targeted system for the reader to understand the scope of the SCE application. It is also helpful to specify the targeted *quality attributes* or *objectives*, such as

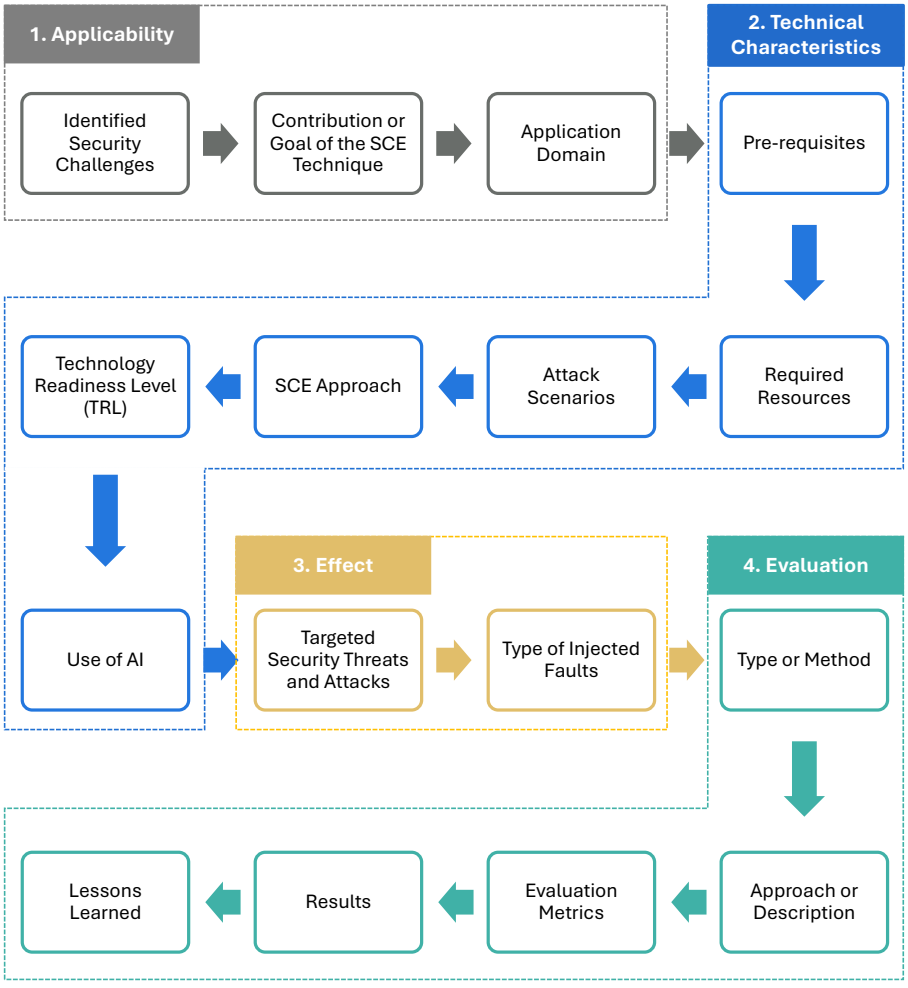

Figure 9: Proposed characteristics for reporting SCE, which form the basis for SCENE guidelines [13] .

performance improvements (e.g., achieving 95% detection accuracy), scalability (e.g., supporting deployment across distributed systems), or usability (e.g., requiring minimal configuration by end-users).

Reporting Contribution/Goal of the Technique

The goal of the technique is to [*strategic goal*] and [*functional goal*]. The technique targets these [*quality objectives*] for this [*targeted system*] and [*use case*].

Application Domain: Defining the application domain helps situate the technique within a specific operational context, guiding users on where it is most effective. Domains can include sectors such as healthcare, finance, transportation, industrial control systems, cloud computing, IoT, or embedded systems. When reporting the application domain, reporters should describe any domain-specific *constraints*. In telecommunications, application domain constraints often revolve around the need for ultra-low latency, high availability, and strict adherence to communication protocols like 5G standards. In automotive, applications often impose constraints related to real-time safety-critical processing, CAN bus integration, and compliance with standards like ISO 26262.

Reporting Application Domain

The technique is designed for [*domain*], where [specific constraints] apply.

5.2. Technical Characteristics

This category describes the implementation-specific and operational aspects of a SCE technique. It outlines the following characteristics.

Pre-requisites: This characteristic outlines the baseline conditions required before the technique can be applied, helping researchers and practitioners understand what must be in place for successful implementation. Prerequisites can be contextual, operational, and/or organizational. Contextual prerequisites include software architecture of the SCE tool, dependencies, integration

requirements, acts or specifics related to regulatory compliance, and critical assets of the system where the technique is to be applied. Operational prerequisites include actions that are required for a successful integration and deployment of the SCE tool. These can be e.g., establishing integration between some specific assets within a specific environment to enable data flow or ensure a correct functioning of the technique. Organizational prerequisites include specific domain knowledge or expertise related to the users of the SCE tool. For example, knowledge in thread modelling is required before applying the developed SCE tool. Pre-requisites should be explicitly reported. This helps define the baseline conditions clearly, enabling accurate setup and minimizing the risk of misinterpretation by future researchers or practitioners.

Reporting Pre-requisites

There exist [*contextual*], [*operational*], and/or [*organizational*] pre-requisites for a successful application of the technique.

Required resources: Reporting on the required resources helps researchers and practitioners in assessing whether the SCE method is feasible in their context. Required resources should outline the technical and infrastructural inputs necessary to implement or replicate the technique. These resources may include hardware (e.g., CPUs, GPU-enabled servers, or embedded devices), test environments and infrastructures (e.g., virtual machines, cloud platforms, or sandbox setups), software and dependencies (e.g., open source software, specific operating systems, libraries, platforms, or system source code), and datasets (e.g., network traffic logs, vulnerability databases, or threat intelligence).

Reporting Required resources

The technique requires [*hardware resource*], [*software resource*], [*infrastructure resource*], and/or [*data resource*].

Attack scenarios: Attack scenarios help stakeholders understand the adversarial conditions the technique is designed and evaluated to withstand and assess whether those conditions align with their own threat landscape. These attack scenarios should report information about the attack goals, attack-defence trees, attack techniques and capabilities, constraints, and/or conditions of the target system. Reports should explain the rationale behind the selection of attack scenarios (e.g., relevance to domain or prevalence in real-world incidents), map them to established threat models, e.g., MITRE ATT&CK, and clearly state assumptions made (e.g., attacker has access to internal logs or can intercept network traffic).

Reporting Attack scenarios

The attack scenarios considered by the technique include [*scenario*] that are selected based on the following [*rationale*] and [*assumption(s)*]. These scenarios are mapped to [*threat model(s)*].

SCE approach: Clearly describing the SCE approach adopted in a study is essential for building confidence in the methodology, ensuring transparency, and enabling reproducibility. This characteristic helps stakeholders understand the nature and structure of the technique, whether it is a novel contribution or adaptation of an existing method. The SCE reports should justify and provide details about the selected approach or process, along with how the adoption of the approach can be integrated into the DevSecOps life-cycle. These details should be self-contained including a detailed explanation of the main activities or steps that are required to conduct the SCE.

Reporting SCE Approach

The technique is based on [*method/process*], selected due to [*rationale*], and consists of these [*steps/activities*].

Technology Readiness Level (TRL): Reporting the TRL of an SCE technique provides readers with a clear understanding of its maturity and proximity to real-world deployment. TRL is a standardized metric that indicates whether a technology is conceptual, validated in a controlled environment, or proven in operational settings. This helps stakeholders assess the applicability of the technique to their own systems and determine the level of investment or adaptation required. Authors should specify the TRL based on an established standards (e.g., [22]), and provide evidence supporting the assessment, such as proof-of-concept results, lab experiments, pilot deployments, or field trials. For instance, a technique tested in a simulated environment with synthetic data may be classified as TRL 4, while one deployed in a live enterprise network with real-time monitoring may reach TRL 7 or higher.

Reporting TRL

The technique is assessed at TRL [*level*], based on [*standard*], supported by [*evidence* such as test results, deployment context, or stakeholder feedback].

Use of AI: When AI is used in an SCE technique, it introduces both powerful capabilities and potential limitations, making it essential to clearly document its role and implementation. Reporting the use of AI helps

stakeholders understand how intelligent components contribute to the technique, whether for detection, simulation, decision support, or adaptive response. Moreover, it allows for critical evaluation of performance, reliability, and bias. Authors should specify the type of AI models used (e.g., supervised learning, reinforcement learning, deep neural networks), the training data (e.g., labelled datasets, synthetic logs), and the configuration details (e.g., hyper-parameters, feature selection methods, model architecture). It is also important to justify the choice of AI techniques by explaining why they are suitable for the problem space (i.e., enhancing the cybersecurity resilience of a domain-specific system) and solution space (i.e., technical and implementation decisions), and to highlight any limitations such as data imbalance, explainability issues, or susceptibility to adversarial inputs. With the increasing adoption of LLMs in software engineering practices [16] and in cybersecurity research [23], it is essential to transparently report how these models are used within an SCE study. This includes specifying the particular LLM employed, the data used to train the model (if not a pre-trained model is used), and whether the model has been fine-tuned or extended using techniques such as Retrieval-Augmented Generation (RAG). Considering the data, authors should report types of data that are useful for conducting SCE experiments. The data depends on the investigated problem space and aim of the SCE study, and might include e.g., targeted system data, common vulnerabilities, common weaknesses, Threat Analysis and Risk Assessment (TARA) reports, cybersecurity standards, and other threat intelligence. In addition, the prompts used in the SCE study should be clearly documented, as different prompting strategies can have a substantial impact on the quality, consistency, and reliability of the generated outputs [17]. Furthermore, it is important to clearly describe the tasks for which LLMs are applied and to explain how common limitations, such as hallucinations, are identified and mitigated. Finally, when integrating AI models into a process, SCE studies should report how existing workflows are adapted or redesigned, including where human-in-the-loop involvement is incorporated and how the quality of AI-generated outputs is assessed or validated.

Reporting Use of AI

The technique uses [*AI model(s)*] trained on [*dataset(s)*], configured with [*parameters* or/and *fine-tuning approach* or/and *prompts*] to perform [*function(s)*]. The choice is motivated by [*rationale*] with limitations including [*limitations*]. The AI-based process is [*fully automated* or *involves human-in-the-loop*]. Finally, the quality of the AI-generated outputs are assessed or validated using [*validation approach*].

5.3. Effect

This category helps in assessing the impact and effectiveness of the SCE technique. This information is essential for stakeholders to evaluate the robustness of their security measures and identify potential vulnerabilities. This category outlines the following characteristics.

Targeted Security Threats and Attacks: Identifying the specific security threats and attacks that the technique aims to mitigate is crucial for assessing its relevance and effectiveness. This allows stakeholders to determine whether the technique addresses the risks most pertinent to their systems. Reporters should report the threats using established threat modelling frameworks such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege).

Reporting Targeted Threats and Attacks

The technique targets [*threat type*] and/or [*attack type*].

Types of Injected Faults: Describing the types of faults the technique is tested against provides insight into its resilience and fault tolerance. This is important for understanding how the technique behaves under stress, misconfiguration, or malicious interference. Faults can include manipulation of system configurations, disabling of security features, data corruption, protocol violations, or resource exhaustion. Beside the type of fault, reporters should report the *execution context* during which the fault was introduced (e.g., compile time, deployment time, or runtime).

Reporting Types of Injected Faults

The technique is tested by injecting [*type of fault*] during [*execution context*].

5.4. Evaluation

Finally, the *Evaluation* of the SCE technique should be comprehensively reported since it helps to understand how well a SCE technique works. By reporting the sub-categories below, authors make their work transparent and credible.

Evaluation Type/Method: Reporting the type or method of evaluation clarifies how the SCE technique is validated. Authors should specify whether the evaluation is analytical (e.g., formal reasoning, mathematical modeling), empirical (e.g., experiments with real-world data or systems), or hybrid (a combination of methods). Moreover, they should explain the rationale behind the chosen method, which helps readers understand why the evaluation method is selected.

Reporting Evaluation Method

The technique is evaluated using [*method: analytical/empirical/both*] that is selected due to [*rationale*].

Evaluation Description: A transparent description of the evaluation approach is essential for enabling reproducibility, critical assessment, and contextual understanding of the SCE technique's validation. Authors should clearly explain what is evaluated, such as resilience to attacks, detection accuracy, fault tolerance, or system recovery. Moreover, authors should detail the specific evaluation design or protocol by describing the experimental setup (e.g., testbed configuration, simulation environment), the data used (e.g., real-world logs, synthetic datasets), and any replication materials provided (e.g., scripts, configuration files, or open-source packages). Authors should justify the choices the made for the evaluation design by linking them to the study's objectives and constraints, and report the evaluation steps in a structured format.

Reporting Evaluation Description

The evaluation of the technique assessed [*aspect*] using [*setup*], selected due to [*rationale*], and conducted with [*tools/data*].

Evaluation Metrics: Evaluation metrics provide objective criteria to measure the effectiveness, efficiency, and/or reliability of an SCE technique. Reporting them clearly allows for consistent comparisons across studies and helps stakeholders understand the trade-off involved. Authors should define each metric used, explain its relevance to the study's goals, describe what it captures, and detail how it is measured. Common metrics include time-to-detect, false positive rate, fault coverage, performance degradation, resource utilization, and system recovery time.

Reporting Evaluation Metrics

Metric [*name*] is used to measure [*aspect*], calculated using [*method*], and selected because [*justification*].

Evaluation Results: Evaluation results provide concrete evidence of how the SCE technique performed under the tested conditions, validating its intended goals. Authors should summarize key findings, explain their significance for system resilience or security assurance [24], and relate them back to the study's objectives. Results should highlight both strengths and limitations, avoiding selective reporting.

Reporting Evaluation Results

The evaluation showed that the SCE technique achieved [*key result(s)*].

Lessons Learned: Lessons learned provide actionable insights beyond raw results. Authors should articulate what is discovered during evaluation (e.g., unexpected system behaviours, methodological challenges, tooling limitations), analyse the pros, cons, and side effects (e.g., system overload, service disruption, and/or various risks) of the techniques used. Moreover, the authors should report why the insights are important for improving practice, and how they can inform future SCE techniques or applications. This reflection adds practical value and supports knowledge transfer across the community.

Reporting Evaluation: Lesson Learned

Lesson learned from the application and evaluation of the technique include [*insight*], [*pros*], [*cons*], and/or [*side effect(s)*].

6. Evaluation of the SCENE Guidelines

In this section, we present and discuss the results of the evaluation of the SCENE guidelines. First, we report feedback on the guidelines from researchers and practitioners. Second, we describe the findings from mapping SCENE to established guidelines. Third, we report the implications to research and practice. Last, we discuss the threats to validity.

6.1. Feedback from Researchers and Practitioners

Figure 10 shows the distribution of the answers of the participants on the closed-ended evaluation questions. The figure shows that the perceptions of the participants on the usefulness, understandability, practicality, and completeness of the SCENE guidelines are positive. Indeed, the majority of the participants agree that the guidelines are useful (87%), understandable (74%), practical (65%), and complete (60%).

The answers to the open-ended evaluation questions are collaboratively analysed and interpreted by three authors of this study. The results are presented in the following sections.

Usefulness. SCENE is considered as highly valuable, particularly for researchers, because it offers a structured and easy-to-follow process for SCE. The inclusion of a comprehensive template is seen as a major strength, ensuring that critical aspects are not overlooked. The participants appreciated the systematic approach and its potential to standardize reporting in this emerging field. However, two concerns were raised.

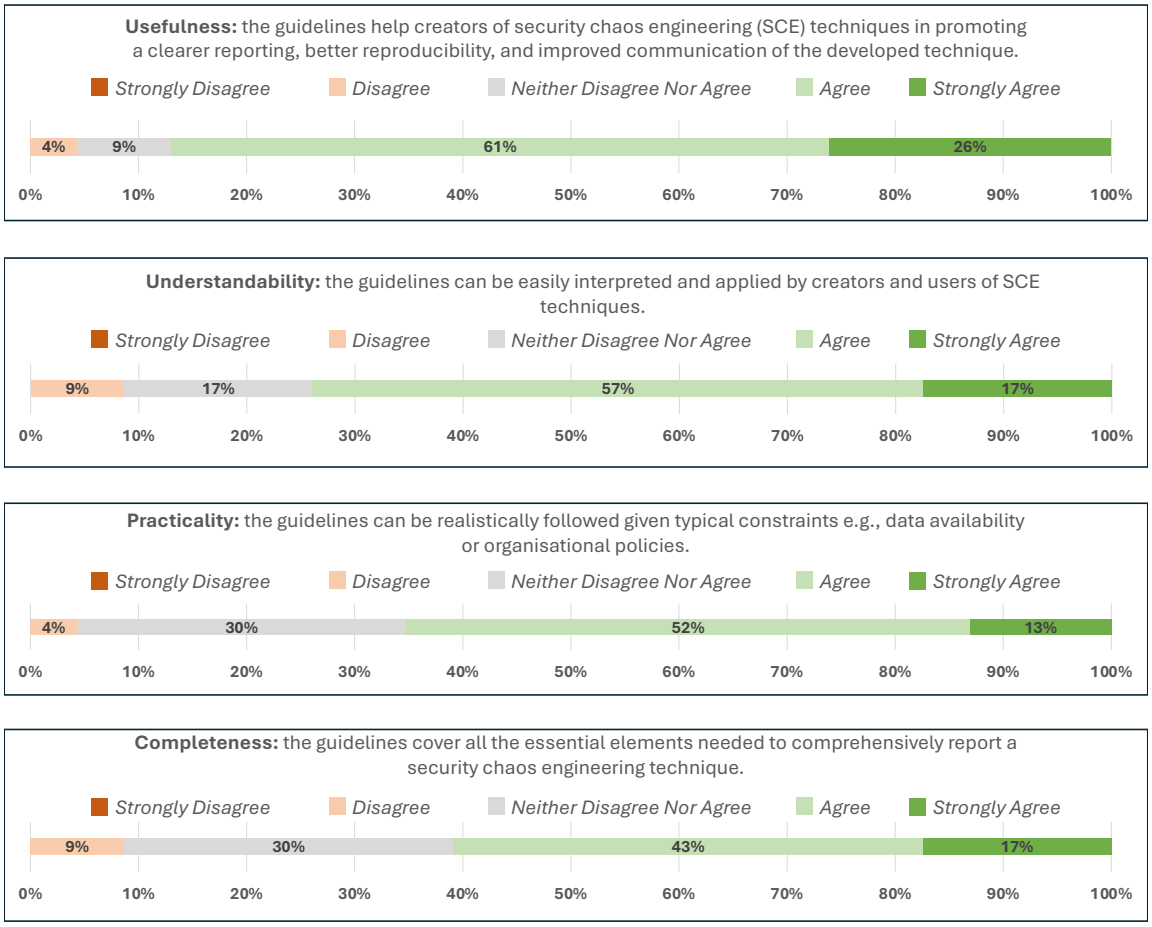

Figure 10: Results of the evaluation of the SCENE guidelines.

First, the placement of application domain in the applicability section was questioned by one participant, as these are viewed as foundational elements that should perhaps come earlier. Second, one participant noted that the evaluation part was not immediately clear, indicating a need for more guidance or explanatory details.

Understandability. SCENE is generally perceived as clearly presented and easy to follow at a high level, especially due to the clear categorization and consistent terminology. However, several comments highlight that while the overall structure is understandable, applying it in practice may be challenging. One participant expressed uncertainty about the appropriate level of abstraction when creating documentation and noted that some terms, such as "*Use of AI*" lack sufficient explanation. The guidelines were described as a good skeleton but needing more details to ensure correct interpretation. Moreover, one participant mentioned that some characteristics overlap or intersect, making it difficult to determine where specific information belongs. For example, prerequisites, required resources, and attack scenarios are closely related, which may lead to confusion when deciding under which SCENE characteristic certain details should be reported. Finally, one participant noted that, for non-researchers, the evaluation method may be difficult to interpret and that the concept of TRL levels is not widely understood.

Practicality. SCENE is seen as potentially very practical because its standardized approach can help practitioners cover critical aspects of system security. The guidelines are considered as realistic to follow in most research and industrial contexts. However, several comments point out that the guidelines currently operate at a high level of abstraction, requiring users to have significant expert knowledge to apply them effectively. For example, knowing which specific fault injection techniques to use in a given context. One participant believes that SCENE itself is not the main limiting factor; rather, the challenge lies in practitioners' willingness to document their ideas thoroughly.

Completeness. SCENE is viewed as providing a solid foundation. Moreover, it is considered as mature and well-scoped at this stage rather than overly complex. However, several comments indicate that it currently

lacks sufficient depth and detail to be considered fully complete. While the structure is appreciated, one participant feels that more specific guidance is needed for each section to avoid ambiguity and ensure consistent interpretation. Suggestions from another participant included adding detailed explanations for all elements, clarifying the intended level of abstraction, and providing concrete examples or sample reports to illustrate how SCENE should be applied in practice.

Interpretation. The feedback on SCENE was positive. Many expressed appreciation for the concept and its potential to bring structure to SCE reporting. The reported shortcomings across all evaluation aspects can largely be explained by the evaluation context and design. Indeed, SCENE was intentionally presented at a conceptual level to keep the evaluation within approximately 15 minutes. Accordingly, evaluators did not have detailed instructions, examples, or full explanation of the guidelines, which naturally led to comments about missing details, ambiguity, and lack of clarity in certain terms or sections. Moreover, without providing practical examples or real-world scenarios in the evaluation material, evaluators might have had to imagine how SCENE would work in practice. This likely amplified concerns about applicability, level of abstraction, and completeness.

6.2. Mapping the SCENE to established software engineering guidelines

The mapping between SCENE and the reporting guidelines of Jedlitschka et al. is presented in Table 9. The results indicate that SCENE is largely compatible with existing methodological guidance. As seen in Table 9 SCENE steps align well with the corresponding items in Jedlitschka et al.'s framework, suggesting that SCENE does not contradict established practices.

Table 10 presents the outcome of comparing SCENE against the issues identified by Kitchenham et al. in their evaluation of Jedlitschka et al.'s guidelines. This analysis shows that SCENE can address several of the gaps noted by Kitchenham et al., particularly in relation to questions that help practitioners assess the applicability of experimental results in real-world settings. For example, SCENE prompts reporting on study context, participant characteristics, and practical relevance, all of which contribute to bridging the gap between academic studies and industrial use.

6.3. Implications for researchers and practitioners

From an industrial practitioner's point of view, the technical characteristics of the tool/framework used in a study are one of the most interesting and important aspects. TRL and related attributes such as reliability and credibility related to the company, organization, or group of individuals behind the tool or technology in question are crucial. For example, it is unlikely that a well-established company will allow their operations to rely on tools with a low TRL, or when the provider of the tool lacks credibility. A case in point is the TRL characteristic that was missing in most papers. Following SCENE in a stringent manner would address a topic that otherwise tends to become a measured perception of technical characteristics such as TRL. From the perspective of practitioners, SCENE is considered to cover areas in SCE study reporting that will aid in implementing SCE techniques and integrating these in daily work flows.

Beyond technical maturity, the explicit reporting of threat coverage has important practical implications. Our SLR indicates that certain threat classes, such as Elevation of Privilege, are rarely targeted in existing SCE studies, despite their relevance for real-world security incidents. For practitioners, such blind spots can lead to a false sense of security if SCE experiments primarily focus on externally observable failures (e.g., availability or data exposure) while internal privilege-related weaknesses remain untested. By requiring the explicit documentation of targeted threats and assumptions, SCENE encourages researchers and practitioners to consciously reflect on which threat classes are addressed and which are omitted, thereby promoting more balanced and comprehensive SCE experimentation over time.

Furthermore, SCENE emphasizes reporting how SCE techniques (including those involving AI) can be embedded within the DevSecOps lifecycle and its associated processes, while explicitly considering human-in-the-loop and the impact of AI usage on workflow integration. This enables practitioners to better anticipate the effort and resources required not only to adopt SCE techniques, but also to effectively integrate them into existing development and security processes.

In addition, Table 10 shows how the guidelines are aligned with the questions highlighted by Kitchenham et al. [18] that relate to industrial practitioners, hence SCENE is considered as complementary to established reporting guidelines, closing the gap between SCE research and practice.

6.4. Threats to Validity

This study is subject to threats to its construct, internal, external, and reliability validity.

6.4.1. Construct Validity

Construct validity concerns the extent to which the collected data accurately reflect what the researchers intended them to represent in the study. The literature review is conducted by involving four search databases, namely IEEE Xplore, ACM Digital Library, Web of Science, and Scopus. Nevertheless, there is a risk of not considering a representative set of relevant studies. To complement the review and ensure broader coverage of relevant studies, we employ the snowballing search approach [40].

Table 9
Mapping SCENE to guidelines for reporting experiments in software engineering

Category	Characteristic	Corresponding Section
Applicability	Identified security challenges	Introduction – Problem statement
	Contribution or goal of the SCE technique	Introduction – Research objective
	Application Domain	Introduction – Context
Technical Characteristics	Pre-requisites	Introduction – Context
	TRL	Introduction – Context (Can also be in Research objectives)
	Attack scenarios	Experiment planning – Experimental unit
	SCE approach	Experiment planning – Design and Procedure
	Required resources	Experiment planning – Procedure and Execution – Preparation
	Use of AI	Experiment planning – Procedure
Effect	Targeted security threats and attacks	Experiment planning – Experimental material
	Type of injected faults	Experiment planning – Experimental material
Evaluation	Type or Method	Experiment planning – Analysis procedure
	Approach or Description	Experiment planning – Analysis procedure
	Evaluation Metrics	Experiment planning – Hypothesis, parameters, and variables
	Results	Analysis – Hypothesis testing and Analysis – Descriptive statistics and Discussion – Evaluation of results and implications
	Lessons learned	Discussion – Lessons learned

Table 10
How the SCENE guidelines complement existing experiment reporting guidelines by addressing practitioner-oriented questions and perspectives.

Question	Addressed by
Is the paper easy to find?	We have created a live SLR with the possibility for researchers to include their SCE papers in the future, which makes finding a relevant SCE study easier for practitioners.
How can the results be used in practice?	SCENE supports this by including the TRL level, the pre-requisites, and the required resources in SCENE.
Is the availability of required support environment clear?	The pre-requisites and the required resources characteristics include reporting on the required tool support and their availability.
Are any technology pre-requisites specified?	SCENE has a dedicated part to report pre-requisites for the technology (SCE) used.
Are the experience or training costs required by development staff defined?	The required resources part of SCENE include the required experience and potential training required to conduct the SCE study.
Are any risks associated with adoption defined?	In the lessons learned characteristic of SCENE, it is explicitly recommended to report side-effects, including the risks of the SCE technology adoption.
Do the results scale to real life?	The TRL level gives an indication for practitioners about the scalability of the results to real-life.
Is the experiment based on concrete examples of use/application or only theoretical models?	SCENE includes characteristics that provide information about the targeted system and use case in which the SCE study takes place. This includes reporting whether the case is from real-life or in an experimental/lab setting.
Is the new approach, technique, or technology well described?	In SCENE, the main technology is SCE, and there is a specific characteristic to describe the used SCE approach.
Does the paper make it clear what commitment is required to adopt the technology?	SCENE stresses the reporting on pre-requisites, TRL, use of AI, and required resources. This makes it easier for a practitioner to decide if the approach is applicable in their context and whether it would require a process change. Reporting on process changes based on an applied technology is out of scope of an SCE paper, and would rather be a stand-alone work.
Are Technology Transfer issues discussed?	In the identification of security challenges, SCENE requires reporting motivations for using SCE to address the cybersecurity challenges.

Discretizing continuous properties, e.g., the measurements of agreement level when evaluating the qualities of the guidelines could pose a threat to construct validity. However, this potential issue was examined using balanced Likert scales and found not to compromise the integrity of the measurements [29].

Part of the SCENE guideline evaluation involved mapping them to established software engineering experiment guidelines. We chose Jedlitschka et al. [12] due to their wide adoption (500+ citations at the time of this study) and focus on both conducting and reporting experiments. To reduce selection bias, we also drew on Kitchenham et al.'s [18] evaluation that identifies gaps in Jedlitscka et al's guidelines and discuss how SCENE addresses them.

6.4.2. Internal Validity

Internal validity concerns efforts made to ensure that possible confounding factors are identified and alleviated. Three researchers (i.e., three authors of this study) conducted the inclusion and exclusion of studies for the systematic literature review. As this process

involves subjective judgment, it poses a potential threat to the reliability of the review. To mitigate this threat, the three researchers performed a quality control of the review by independently analysing a sample of 30 studies i.e., 10.5% of the total retrieved studies. The analysis resulted in an inter-rater agreement coefficient (Fleiss's kappa) of 0.8 which indicates almost perfect agreement according to [20]. The differences in the considered studies (i.e., disagreements) are discussed and resulted in a clarification of the inclusion and exclusion criteria.

The level of expertise and experience in cybersecurity of the evaluators may affect their comprehension and evaluation of the framework. To account for this, we decided to get feedback from individuals working in both academia and industry. These contacted evaluators have a diverse range of expertise e.g., high, medium, and low. This approach was intended to enhance realism, recognizing that not all the evaluators possess advanced knowledge in cybersecurity. Moreover, involving evaluators with varying backgrounds enables the collection of more diverse and meaningful feedback regarding the framework's understandability, usefulness, practicality, and completeness. These feedback might be overlooked if only experts were consulted.

The way the framework is presented (Figure 9) may have influenced the understanding and subsequent evaluation thereof. To mitigate this, we ensured that the presentation is clear and well-structured. Additionally, the feedback on the framework's understandability is generally positive, suggesting that any potential impact from presentation-related factors is minimal.

6.4.3. External Validity

External validity concerns the extent to which results of a study can be generalized. Factors such as the clarity, scope, and complexity of evaluation questions may have limited the generalizability of the results. To mitigate this threat, we encourage the replication of the study. Additionally, the evaluators were recruited through existing contacts and collaborators recruited via a convenience sampling method. While the number of evaluators may be considered limited, this is not viewed as a significant threat, as the primary objective of the evaluation was to gather preliminary qualitative insights rather than to generalize findings across a broader population.

A second potential threat arises from our decision not to collect demographic or detailed professional background information about participants. This choice was intentional to preserve anonymity, reduce response biases, and align with ethical considerations related to privacy and data protection. However, this limits the extent to which the feedback can be generalized to the cybersecurity community. Without more granular demographic or role-specific data, it is not possible to determine whether the perspectives gathered are representative of particular domains, industries, or organizational contexts. Consequently, although participants possessed substantial cybersecurity experience, the lack of detailed background information may reduce the ability to generalize the findings to broader populations or to compare responses across different practitioner profiles.

Traditional SLRs face the risk of quickly becoming obsolete over time. To mitigate this risk, we aimed for and created a live SLR. Indeed, by continuously incorporating newly published studies, the live SLR enhances external validity through improved generalizability and relevance to current research and practice.

6.4.4. Reliability

Reliability refers to the extent to which a study's operations can be replicated by other researchers, yielding consistent results. To support reproducibility, we provide a detailed account of the approach used in conducting the systematic literature review. Additionally, we describe the evaluation process of the framework, including the contextual introduction and the evaluation questions. These comprehensive details are intended to facilitate replication of the study under comparable conditions.

7. Conclusion

Security Chaos Engineering (SCE) is an approach for enhancing the resilience of systems. Several SCE techniques are proposed in literature, but their diversity underscores the necessity for a collective analysis to uncover valuable practices and potential research opportunities. Thus, we conduct a systematic review and collect 414 studies, from which we identify 19 SCE techniques for analysis. The results show a limited and unsystematic investigation of SCE by the community. Accordingly, we develop SCENE, a set of guidelines designed to systematically report SCE practices, aiming to support the clarity, consistency, and reproducibility thereof. SCENE is evaluated by cybersecurity practitioners and active researchers in the field, and is mapped to established methodological guidelines. The results indicates that SCENE is perceived positive in terms of usefulness, understandability, practicality, and completeness. SCENE is also found to complement established experimental reporting guidelines and bridge the gap between academic studies and industrial use.

7.1. Future Work

While the SCENE guidelines are intentionally defined at a foundational level, an important direction for future work is the systematic increase of their granularity as the empirical maturity of the field advances. In particular, the current formulation of SCENE does

not explicitly differentiate between security chaos engineering practices applied at different stages of system maturity (e.g., early exploratory systems versus more operational and mature deployments), nor does it provide domain-specific refinements for settings such as cloud-based systems or IoT environments.

Building on the insights from the live SLR, future work will leverage the continuously updated body of empirical evidence to identify systematic variations in how SCE techniques are reported, evaluated, and applied across maturity levels and domains. As the number of studies evaluating SCE in operational and production environments grows, this evidence will enable the derivation of a refined second iteration of the guidelines with more fine-grained reporting guidance grounded in observed practice rather than speculation. Furthermore, emerging trends such as the increasing integration of AI-driven mechanisms in SCE will be systematically analysed to assess their implications for reporting, evaluation, and reproducibility of SCE techniques.

Acknowledgment

This research was supported by Vinnova, Sweden's Innovation Agency. Project's number: 2024-01718.

CRediT authorship contribution statement

Rodi Jolak : Conceptualisation of this study, Methodology, Guidelines, Evaluation. **Mazen Mohamad :** Conceptualisation of this study, Methodology, Guidelines, Evaluation. **Ramana Reddy Avula :** Conceptualisation of this study, Methodology, Guidelines, Live SLR. **Jason Meek :** Conceptualisation of this study, Methodology, Guidelines, Evaluation. **Alexander Åström :** Conceptualisation of this study, Methodology, Guidelines, Evaluation.

References

[1] Bailey, T., Marchione, P., Swartz, P., Salih, R., Clark, M.R., Denz, R., 2022. Measuring Resiliency of System of Systems using Chaos Engineering Experiments, in: Proceedings of SPIE - The International Society for Optical Engineering.

[2] Basiri, A., Behnam, N., De Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., Rosenthal, C., 2016. Chaos engineering. IEEE Software 33, 35–41.

[3] Bedoya, M., Palacios, S., Diaz-López, D., Nespoli, P., Laverde, E., Suárez, S., 2023. Securing cloud-based military systems with Security Chaos Engineering and Artificial Intelligence, in: ACM International Conference Proceeding Series.

[4] Bedoya, M., Palacios, S., Díaz-López, D., Laverde, E., Nespoli, P., 2024. Enhancing DevSecOps practice with Large Language Models and Security Chaos Engineering. International Journal of Information Security 23, 3765 – 3788.

[5] Belyaeva, S., Yanovich, Y., 2023. Infrastructure Security Checking Service Based on Chaos Engineering Method, in: 2023 18th International Symposium on Problems of Redundancy in Information and Control Systems, REDUNDANCY 2023, pp. 100 – 105.

[6] Boyatzis, R.E., 1998. Transforming qualitative information: Thematic analysis and code development. sage.

[7] Creswell, J.W., Creswell, J.D., 2017. Research design: Qualitative, quantitative, and mixed methods approaches. Sage publications.

[8] Dedousis, P., Stergiopoulos, G., Arampatzis, G., Gritzalis, D., 2023. Enhancing Operational Resilience of Critical Infrastructure Processes Through Chaos Engineering. IEEE Access 11, 106172 – 106189.

[9] Devi, D.P., 2023. Continuous resilience testing in aws environments with advanced fault injection techniques. Int. Journal of Information Technology and Computer Engineering 11, 199–217.

[10] Fogli, M., Giannelli, C., Poltronieri, F., Stefanelli, C., Tortonesi, M., 2024. Chaos Engineering for Resilience Assessment of Digital Twins. IEEE Transactions on Industrial Informatics , 1134 – 1143.

[11] Hölzl, M., Rauschmayer, A., Wirsing, M., 2008. Engineering of software-intensive systems: State of the art and research challenges. Software-Intensive Systems and New Computing Paradigms: Challenges and Visions , 1–44.

[12] Jedlitschka, A., Ciolkowski, M., Pfahl, D., 2008. Reporting experiments in software engineering, in: Guide to advanced empirical software engineering. Springer, pp. 201–228.

[13] Jolak, R., Avula, R.R., Mohamad, M., 2026. SCENE Guidelines and Live SLR for Security Chaos Engineering. URL: https://github.com/rodijolak/SCENE-Guidelines.

[14] K. A. Torkura, M. I. H. Sukmana, T. Strauss, H. Graupner, F. Cheng, C. Meinel, 2018. CSBAuditor: Proactive Security Risk Analysis for Cloud Storage Broker Systems, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pp. 1–10. doi:10.1109/NCA.2018.8548329. journal Abbreviation: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA).

[15] Kalka, W., Szydlo, T., 2024. μ Chaos: Moving Chaos Engineering to IoT Devices, in: Franco, L., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (Eds.), Computational Science – ICCS 2024, Springer Nature Switzerland, Cham. pp. 239–254. doi:10.1007/978-3-031-63783-4_18.

[16] Khojah, R., Mohamad, M., Leitner, P., de Oliveira Neto, F.G., 2024. Beyond code generation: An observational study of chatgpt usage in software engineering practice. Proc. ACM Softw. Eng. 1. URL: https://doi.org/10.1145/3660788.

[17] Khojah, R., de Oliveira Neto, F.G., Mohamad, M., Leitner, P., 2025. The impact of prompt programming on function-level code generation. IEEE Transactions on Software Engineering 51, 2381–2395.

[18] Kitchenham, B., Al-Khilidar, H., Babar, M.A., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., Zhu, L., 2008. Evaluating guidelines for reporting empirical software engineering studies. Empirical Software Engineering 13, 97–121.

[19] Kitchenham, B., et al., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-12007. Keele University.

[20] Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. biometrics , 159–174.

[21] Lewis, J., Wang, C., 2019. Chaos engineering: New approaches to security. A Rain Capital Research Note .

[22] Mankins, J.C., et al., 1995. Technology readiness levels. White Paper, April 6, 1995.

[23] Mert Gültekin, F., Lilja, O., Khojah, R., Wohlrab, R., Damschen, M., Mohamad, M., 2025. Leveraging large language models for cybersecurity risk assessment–a case from forestry cyber-physical systems. arXiv e-prints 2510 .
[24] Mohamad, M., Steghöfer, J.P., Knauss, E., Scandariato, R., 2024. Managing security evidence in safety-critical organizations. Journal of Systems and Software 214, 112082. doi:https://doi.org/10.1016/j.jss.2024.112082.
[25] Naqvi, M.A., Malik, S., Astekin, M., Moonen, L., 2022. On Evaluating Self-Adaptive and Self-Healing Systems using Chaos Engineering, in: Proceedings - 2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems, pp. 1 – 10.
[26] Owotogbe, J., Kumara, I., Heuvel, W.J.V.D., Tamburri, D.A., 2024. Chaos engineering: A multi-vocal literature review. arXiv preprint arXiv:2412.01416 .
[27] Palacios Chavarro, S., Nespoli, P., Díaz-López, D., Niño Roa, Y., 2023. On the Way to Automatic Exploitation of Vulnerabilities and Validation of Systems Security through Security Chaos Engineering. Big Data and Cognitive Computing 7.
[28] Pierce, T., Schanck, J., Groeger, A., Salih, R., Clark, M.R., 2021. Chaos engineering experiments in middleware systems using targeted network degradation and automatic fault injection, in: Proceedings of SPIE - The International Society for Optical Engineering.
[29] Ray, J.J., 1982. The construct validity of balanced likert scales. The Journal of Social Psychology 118, 141–142.
[30] Rosenthal, C., Jones, N., 2020. Chaos engineering. O'Reilly Media.
[31] S. Singh, C. H. Muntean, S. Gupta, 2024. Boosting Microservice Resilience: An Evaluation of Istio's Impact on Kubernetes Clusters Under Chaos, in: 2024 9th International Conference on Fog and Mobile Edge Computing (FMEC), pp. 245–252. doi:10.1109/FMEC62297.2024.10710237. journal Abbreviation: 2024 9th International Conference on Fog and Mobile Edge Computing.
[32] Shortridge, K., Rinehart, A., 2023. Security chaos engineering. " O'Reilly Media, Inc.".
[33] Singer, J., Vinson, N.G., 2003. Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering 28, 1171–1180.
[34] Siwach, G., Haridas, A., Chinni, N., 2022. Evaluating operational readiness using chaos engineering simulations on Kubernetes architecture in Big Data, in: 2022 International Conference on Smart Applications, Communications and Networking, SmartNets 2022.
[35] Soldani, J., Brogi, A., 2021. Automated Generation of Configurable Cloud-Native Chaos Testbeds. Communications in Computer and Information Science 1462, 101 – 108.
[36] for Standardization, I.O., 2011. ISO/IEC/IEEE 42010:2011(E). International Standard for Systems and Software Engineering – Architectural description. Standard. International Organization for Standardization. Geneva.
[37] Tian-yang, G., Yin-Sheng, S., You-yuan, F., 2010. Research on software security testing. International Journal of Computer and Information Engineering 4, 1446–1450.
[38] Torkura, K.A., Sukmana, M., Cheng, F., Meinel, C., 2021. Continuous auditing and threat detection in multi-cloud infrastructure. Comput. Secur. 102.
[39] Torkura, K.A., Sukmana, M.I.H., Cheng, F., Meinel, C., 2020. CloudStrike: Chaos Engineering for Security and Resiliency in Cloud Infrastructure. IEEE Access 8, 123044 – 123060.
[40] Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th international conference on evaluation and assessment in software engineering, pp. 1–10.
[41] Zavalyshyn, I., Given–Wilson, T., Legay, A., Sadre, R., Rivière, E., 2021. Chaos Duck: A Tool for Automatic IoT Software Fault-Tolerance Analysis, in: 2021 40th International Symposium on Reliable Distributed Systems (SRDS), pp. 46–55. ISSN: 2575-8462.

Rodi Jolak is a senior researcher at RISE, the Research Institutes of Sweden and an adjunct professor of software engineering at Mid Sweden University. Rodi received a Ph.D. in computer science and engineering from the University of Gothenburg, 2020. His research interests include software engineering (SE), secure SE, artificial intelligence (AI) for SE, and SE for AI. Check for more at https://rodijolak.com

Mazen Mohamad is a senior researcher at RISE, the Research Institutes of Sweden and a lecturer at Chalmers University of Technology. He holds a Ph.D. degree in software engineering from the University of Gothenburg. His research focuses on security assurance, combined safety and security analysis, AI in software engineering, and AI for cybersecurity. Contact him at mazen.mohamad@ri.se or www.mazenm.com.

Ramana R. Avula received the dual B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Madras, India, in 2015, and the Ph.D. degree in electrical engineering from the KTH, Sweden, in 2023. He is currently a senior researcher at RISE, the Research Institutes of Sweden. His research interests lie in statistical signal processing, with an emphasis on enhancing the safety, security, and reliability of autonomous systems.

Jason Meek is a senior cybersecurity engineer working at Volvo Group Trucks Technology and Industrial Division (TTI). He holds a MSc. degree in electrical engineering from Chalmers University of Technology. His main focus is on research and development within product cybersecurity in the connectivity domain.

Alexander Åström is a cybersecurity specialist working as a Consultant in the Automotive domain, focusing on both development and research. He has an MSc in Electrical Engineering from Chalmers University of Technology with a specialization in Computer- and Programming systems. He currently works as a cybersecurity architect within Volvo Group Trucks Technology and Industrial Division (TTI).