Available online at www.ijournalse.org

Emerging Science Journal

(ISSN: 2610-9182)

Vol. 7, No. 3, June, 2023

Review Article

Overview on Case Study Penetration Testing Models Evaluation

Ahmad S. Al-Ahmad [1*], Hasan Kahtan [2], Yehia I. Alzoubi [1]

[1] *College of Business Administration, American University of the Middle East, Egaila 54200, Kuwait.*

[2] *Cardiff School of Technologies, Cardiff Metropolitan University, CF5 2YB Cardiff, United Kingdom.*

Abstract

Model evaluation is a cornerstone of scientific research as it represents the findings' accuracy and model performance. A case study is commonly used in evaluating software engineering models. Due to criticism in terms of generalization from a single case study and testers, deciding on the number of case studies used for evaluation and the number of testers has been one of the researchers' challenges. Multiple case studies with multiple testers can be difficult in some domains, such as penetration testing, due to the complexity and time needed to prepare test cases. This study aims to review the literature and examine the evaluation methods used pertaining to the number of case studies and testers involved. This study is beneficial for researchers, students, and penetration testers as it provides case study design steps that are useful to determine the appropriate number of test cases and testers required. The paper's findings and novelty highlight that a single case study with a single tester is enough to evaluate a model. It also strikes a balance between what is enough for the evaluation and the need to reduce criticisms of a single case study by using two case studies with a single tester.

Keywords:

Penetration Testing;
Model Evaluation;
Tester; Case Study;
Software Engineering.

Article History:

Received:	20	September	2022
Revised:	26	January	2023
Accepted:	09	March	2023
Available online:	14	May	2023

1- Introduction

Researchers tend to use models to represent their ideas, theories, guidelines, procedures, and findings [1–4]. These models have to be evaluated to determine if they are effective by utilizing one of the evaluation methods [5, 6]. The research domain consists of several evaluation methods, and among the models' evaluation methods is the case study [7, 8]. The case study is one of the most common approaches used in the software engineering domain [9–11] and is generally used in the computer domain [12, 13]. The case study must be well designed in order to provide generalizations for the results of the evaluation [14-16]. The case study design must comprise case study selection and preparation [14, 17-19]. Case study design for new technologies that integrate cloud computing along with other technologies like mobile, routers, sensors, and other IoT devices is more complex than designing case studies for simpler technologies [20, 21].

In 2002, penetration testing was defined as the method of locating open doors in information systems that can be exploited by attackers [22]. Researchers are currently attempting to redefine and improve penetration testing by describing it as a post-deployment vulnerability assessment task that is carried out as an isolated test process in a manual and even ad hoc fashion [23–28]. The main characteristics of penetration testing for security vulnerabilities are flexibility and scalability. The complexity of penetration tests, technology modeling, implementation, and configuration of the testing environment makes designing an evaluation case study a difficult task. For example, the complexity of designing the architecture, implementation, and deployment environment for fog computing and mobile cloud computing applications necessitates configuring and making the cloud available via mobile devices [29–36]. To summarize, the complexity of the architecture of the application under test complicates penetration testing and necessitates expertise in several domains.

* **CONTACT**: ahmad.alahmad@aum.edu.kw

DOI: http://dx.doi.org/10.28991/ESJ-2023-07-03-025

Among the challenging decisions that case study evaluation needs to consider is determining the number of case studies and the number of testers needed to evaluate the testing model [37, 38]. Since most research is criticized for generalization, researchers are attempting to create case studies that will assist them in generalizing their findings [7]. Therefore, researchers frequently refer to relevant, well-known articles as criteria for determining the number of case studies and testers to employ in their model evaluation [39]. During their Ph.D. journey, the authors spent a significant amount of time analyzing their models. A large portion of this time was spent trying to establish the reasons for the generalization critiques they received as a result of using a single case study with a single tester. The primary objectives of this work are as follows: (i) Provide design steps for using a case study in evaluating penetration testing models. (ii) Review related research in terms of the number of case studies. (iii) Review related research in terms of the number of testers (participants) used. (iv) Discuss the recommended number of case studies and number of testers when conducting penetration testing model evaluation. (v) Establish the fundamentals for future complex model evaluation guidelines.

In this work, we refer to the penetration tester who is taking part in the model evaluation as either a participant or a tester. Furthermore, in this context, the penetration testing model refers to the methodologies, procedures, rules, frameworks, and tools used to enhance and improve the penetration testing process in the domain of software engineering, specifically software security testing. Moreover, the case study concept here refers to the case study, which is also known as an observational study or field research in other literature. Case studies, in particular, are empirical investigations that examine a contemporary event in depth and within its context [8].

This study answers mainly the following five research questions. RQ1 (what are the steps to design penetration testing model evaluation using a case study?). RQ2 (what is the number of test cases used in the penetration testing model evaluation?). RQ3 (what is the number of testers used in the penetration testing model evaluation?). RQ4 (how many test cases should be used to evaluate the penetration testing model?). RQ5 (how many testers will be needed to assess the penetration testing model?). While striking a balance between complexity, time, and the results generalizability. The findings of this paper provide case study design steps for the researcher to direct their evaluation of penetration testing models. This may reduce the time and complexity of the evaluation process by helping the researcher to determine the optimal number of case studies and testers.

This work contributes by recommending case study design steps based on literature to help researchers, students, and practitioners in the field of penetration testing throughout the evaluation process. This was accomplished by adhering to a well-structured and approved framework in terms of the number of implemented case studies and a sufficient number of testers. The answers to the aforementioned research questions will aid in the development of a procedural process design for evaluating the penetration testing model using a case study. Furthermore, it aids in establishing the number of case studies and testers required to evaluate the penetration testing model while taking into account the complexity of the testing and delivering generalizable conclusions. This study is a work of art that answers crucial issues and serves as a guideline for researchers and students when designing case studies. This paper's innovation lies in its simplicity and comprehensiveness. At the same time, this paper presents many perspectives and criticisms of research that uses few case studies or a small number of testers, which might be useful in enlightening researchers on additional facts that they should consider and justify while doing their research. This work paves the way for future research and development in areas such as software development, implementation, and maintenance models.

2- Background and Related Work

2-1- Penetration Testing

This section presents a deeper background on penetration testing in order to illustrate the benefits behind this type of testing, discusses the main processes, highlights real scenarios where penetration testing can be helpful, and provides an introduction to the penetration testing model. This section can be considered as a bird's view of penetration testing, which may help in gaining some insights into the current state-of-the-art practice in this domain. Penetration testing has been defined by Engebretson [40] as the tests that are made to discover hidden vulnerabilities by examining and exploiting computer systems (network and software) with the intention of enhancing the security of the systems under test. In other words, in penetration testing, testers are trying to mimic the act of hackers in order to find the weak points of the system under test and report these weak points. These reports are then used by the system administrator and developers to overcome these weak points [41].

As presented in the discussion, penetration testing can be presented as a reverse engineering process of the software and network in order to find what to test, how to exploit security vulnerabilities, and report the results. This reverse process engineering can be conducted using the source code (white box), or it can be done using the executable software and the network implementation (black box), while in some cases it can be a mixed method where only part of the code is available (gray box). The white box, black box, and gray box represent the main types of penetration testing [42].

Penetration testing is trying to find vulnerabilities before hackers do [43]. This will give the software and network engineers an advantage over hackers to fix the penetration testing reposted issue before it is exploited during malicious attacks. Another advantage of penetration testing is that it is well-structured and clear for testers through the steps that are presented in models [44]. This advantage also makes it subject to automation, which can be helpful when implemented on large-scale heterogeneous information systems [45].

Penetration testing can help software engineering in many ways. For example, penetration testing can be used to test the system against certain types of attacks, such as denial of service attacks [46], SQL injection attacks [47], and cross-site scripting attacks [48]. Where the tester starts by analyzing the computer system under test, generating a set of test cases, selecting the important test cases, executing those test cases, and reporting the results [29, 49–52].

2-2- Model Evaluation: Related Works

Model evaluation is one of the most challenging tasks of research as it is complex, time-consuming, and requires extensive effort [53]. The researchers have certain metrics to be evaluated within the domain of the research, like technical requirements, generally accepted standards, usability, and complexity. Along with the domain metrics, researchers need to meet the generalization criteria that confirm that their results can be generalized [14, 18, 54–59]. Among the most common methods to be used in the model's evaluation is the case study evaluation method [14, 58, 59].

Researchers who apply case studies for model evaluations normally tend to use previous research methods of evaluation to support and guide model evaluations or use case study guidelines and frameworks. Certain studies were published to provide guidelines and frameworks to support the researchers during the evaluation phase in multiple domains (e.g., software engineering, software testing, and other domains). Table 1 presents a sample of works that used or proposed a method, framework, and guideline for using case studies in research. There are also a huge number of research publications that use case studies. Table 2 presents a sample of research publications that use case studies.

Table 1. Method, framework, and guideline to use case study in research study

	Reference	Domain	Title
1	[60]	Software engineering	Case studies for software engineers.
2	[61]		Can you trust a single data source exploratory software engineering case study?
3	[62]		Qualitative methods in empirical studies of software engineering.
4	[58]		Scaling up case study research to real-world software practice.
5	[8]	General	Case study research: Design and methods.
6	[63]		Five misunderstandings about case-study research.
7	[64]		Systematic case study research: A practice-oriented introduction to building an evidence base for counselling and psychotherapy.
8	[65]		Enhancing the scientific credibility of single-case intervention research: Randomization to the rescue.
9	[66]		Types of case study work: A conceptual framework for case-based research.
10	[67]		Single case study documentation.
11	[68]	Business	Strengths and weaknesses of business research methodologies: Two disparate case studies.
12	[69]	Sport	Single-case research methods in sport and exercise psychology.

Table 2. Sample of research publications used case study

	Reference	Domain	Title
1	[70]	Software Testing	Effect of test set minimization on fault detection effectiveness.
2	[71]		Managing crowd sourced software testing: A case study-based insight on the challenges of a crowdsourcing intermediary.
3	[72]		Internet banking security management through trust management.
4	[73]		Improving software security with precise static and runtime analysis.
5	[74]		Automated runtime testing of web services.
6	[75]		An evolutionary approach for system testing of android applications.
7	[76]		Utilizing output in web application server-side testing.
8	[77]	Software Engineering	Two's company, three's a crowd: A case study of crowdsourcing software development.
9	[78]		A longitudinal case study of an emerging software ecosystem: Implications for practice and theory.
10	[79]		Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study.
11	[80]	Penetration Testing	Variability testing in the wild: The Drupal case study.
12	[81]		Improving security and privacy of integrated web applications.
13	[73]		Improving software security with precise static and runtime analysis.
14	[82]		A hybrid framework for the systematic detection of software security vulnerabilities in source code.
15	[83]		Cybersecurity testing and intrusion detection for cyber-physical power systems.
16	[84]		Advanced automated web application vulnerability analysis.
17	[37]	Business	Designing and conducting case studies in international business.
18	[85]	Social	Rey: An intensive single case study of a probation youth with immigrant background participating in wraparound Santa Cruz.
19	[86]		A case study of extensive reading with an unmotivated l2 reader.
20	[87]	Health	Is our food safe? an assessment: On the European union food safety policy, concerning the safety of meat and animal-derived food products in the EU.

As shown in Table 1, many studies have proposed guidelines or frameworks that researchers can adapt to their research. For instance, Perry et al. [60] proposed guidelines for software engineering research when using the case study. Ghauri [37] proposed guidelines that can be adapted to international business domain research when using the case study. Similarly, Yin [8] has discussed in detail how to use case studies in research. Flyvbjerg [63] discussed the general misunderstanding of case studies. McLeod & Elliott [64], Zivkovic [68], and Kratochwill et al. [67] are researchers who discussed case studies in the research domain.

Similarly, Table 2 shows multiple studies in certain related domains that used a case study approach in the evaluation process of their proposed models. For instance, Zhou [81], Livshits [73], Hanna [82], and Pan [83] have used case studies in the penetration testing domain for model evaluation. Meanwhile, Alshahwan [76], Mahmood [75], Koskosas and Koskosa [72], Livshits [73], and Ramollari [74] have used case studies in the software testing domain. Hanssen [78] and Da Silva et al. [79] have used case studies in the domain of software engineering.

2-3- Research Methodology

There are a large number of studies that have discussed the case study approach and used case studies in model evaluation. Therefore, in order to meet the objectives of this paper, evidence-based practice was used as proposed by Bastian et al. [88]. In order to collect evidence, this paper has conducted journal searches, database queries, and Internet-based searches similar to those conducted by Sana and Li [89], Ali et al. [90], Alzoubi et al. [91], and Al-Ahmad et al. [92].

To adopt evidence-based practical and theoretical perspectives, researchers, especially in the penetration testing domain, rely on solid empirical evidence that reports methodologies, frameworks, models, and practices. Systematic literature review (SLR) is one of the most suitable methods to encourage evidence-based security practices [93–98]. Researchers followed the SLR rules proposed by Kitchenham & Charters [56] and Keele & Bell [99]. Furthermore, the techniques and rules portrayed by [30, 31, 54, 55, 93, 99–101] were followed by the SLR, i.e., analysis, screening, exclusion, and inclusion. These phases, along with the main outcomes, are shown in Figure 1.

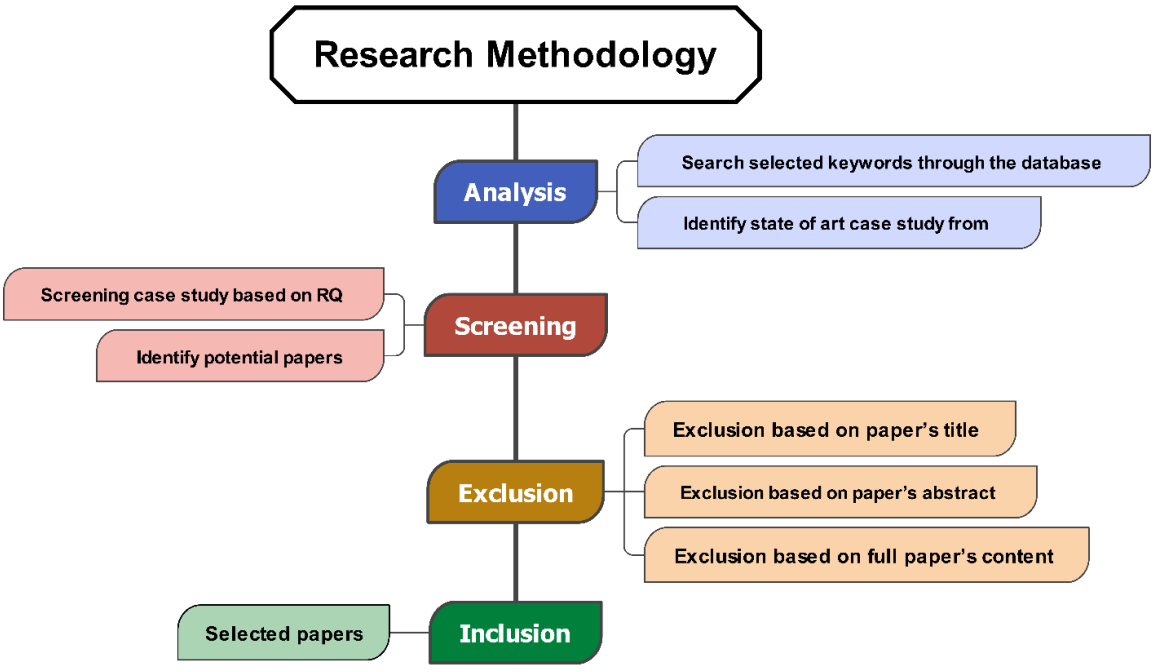

Figure 1. Research Methodology Phases

During the analysis phase, online databases for the literature review were selected. These databases are Scopus, EBSCO, Springer, Emerald, Wiley, MDPI, Taylor, Sage, IGI, IOS, Google Scholar, ACM Digital Library, and Google Search for multiple university libraries. The filtering tools provided by these databases have been used for each study to limit study outcomes [102]. Initial reading of the abstract and the title of the papers was conducted [103, 104]. Then the chosen papers were fully read to sort out irrelevant papers [105].

This research study emphasizes the importance of case studies in model evaluation, certain studies, guidelines for using a case study in model evaluation from multiple domains, and previous research that used case studies in model evaluation by selecting a set of relevant search terms and combining them using "AND" and "OR" Boolean operators. The selected search terms for this study are "Case study", "Evaluation", "Penetration testing", "Model", "Framework", "Methodology", "Best practice", "Generalization", "Software testing", "Software engineering", and "Tester".

During the screening phase, academic papers written in English and listed in reputable and high-quality journals and conferences published from 1995 to 2021 were selected. More than 700 articles published in top-ranked peer-reviewed journals and conferences were found using search terms. During the exclusion phase, 540 were excluded after reading the title and abstract. Only 160 papers were selected based on the review's objectives. After carefully reading the manuscripts of the list of papers selected previously, this study chose the 82 most relevant research articles during the inclusion phase. That showed a lot of consideration in the case study and penetration testing-related research areas.

3- Penetration Testing Model Evaluation Using Case Study

3-1- Case Study Design Steps

Multiple researchers tried to build models to use penetration testing in multiple types of applications [24, 106, 107]. They tried to use penetration to test web applications, mobile applications, desktop applications, web services, and databases using models and frameworks to find hidden vulnerabilities that may be used by attackers to harm applications, interrupt systems, and steal data [106, 108–113]. These models need to be evaluated to prove their efficiency and effectiveness in finding these hidden vulnerabilities. A case study is one of the most common methods used previously in penetration testing model evaluation; for example, it was used by Sánchez et al. [80], Zhou [81], Livshits [73], Hanna [82], Pan [83], and Doupé [84].

Previous case study practices disregarded the selection of the number of case studies and the number of testers for the case study. Therefore, due to the exclusivity of penetration testing and the increasing complexity of the application domain under test, as discussed earlier, we found that there is a need to have a specific case study reference for the penetration testing model evaluation. This reference should provide a comprehensive grounding to help the researcher save time in determining the number of case studies and the number of testers while meeting the needs of generalization. Also, it should take into consideration the penetration testing complexity and time for researchers, still keeping the research quality acceptable.

The proposed case study design steps shown in Figure 2 are trying to answer RQ1 (what are the steps to design penetration testing model evaluation using a case study?). This design is an adaptation of the previously proposed case study design models, frameworks, methods, and guidelines published in highly ranked journals and used by a large number of researchers (e.g., [7, 12, 14, 18, 38, 58, 114–116]). This proposed design simplifies the penetration testing process as it is highly abstracted and presents only the main process for designing a case study for penetration model evaluation. This design starts with selecting the number of case studies and the number of testers. The design also needs to select testbed applications for the two case studies. Then, prepare the application environment. The single penetration tester will be chosen as part of the data collection design in preparation for data collection at the intended research for which the proposed design will be used.

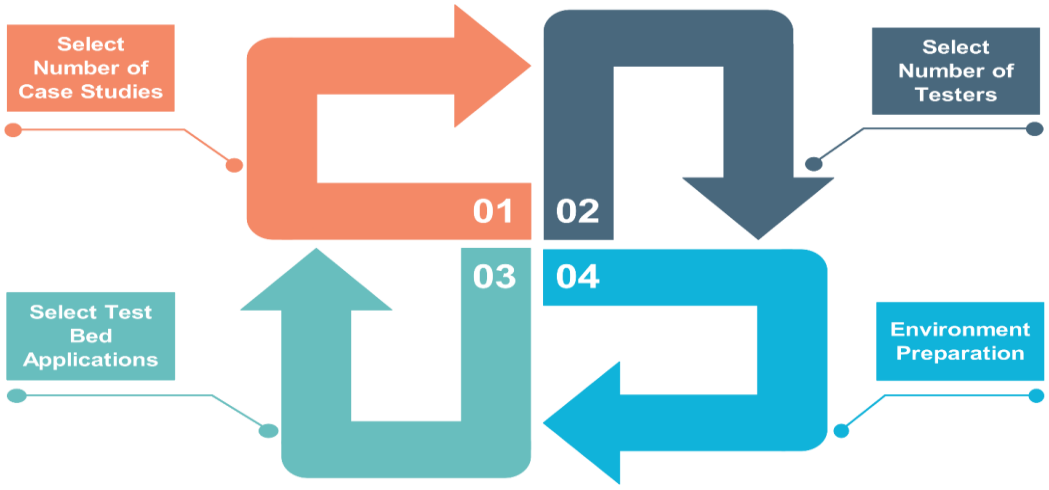

Figure 2. Case study design steps

3-2- Number of Case Studies

This section will review the literature on previous penetration testing model evaluations to answer the second research question RQ2 (what is the number of test cases used in the penetration testing model evaluation?). By the end of this section, a discussion is provided to conclude and recommend the number of case studies to be used in evaluating penetration testing, which answers RQ3 (what is the number of testers used in the penetration testing model evaluation?).

The single-case study approach has been applied both in social science in general and in software engineering in particular. A single case study with one tester is sufficient for generalization [37, 60, 63]. It also helps to make the case more recognizable [64] and may be used to endorse a model or hypothesis contest [68]. Additionally, a single case study

can be the basis for significant explanations and generalizations [114]. The single case study approach is not limited to the social sciences but has also been applied in software engineering, software testing, security testing, and penetration testing research. It has been used previously in software engineering research as it provides more flexible procedures and patterns of working that allow the researcher to find out how different issues hold different significance and focus on research identification and interpretation [62, 70, 77–79]. The single case study approach has also been used in software testing research as it provides a deeper understanding and in-depth insight into important issues [71, 80], in security testing as it provides a rich description of the case study [72, 74], and in penetration testing [82]. The single case study has been used by research that uses real-world software testing case studies in order to efficiently analyze and control the results, as it produces large results [58].

However, common criticisms of a single case study include generalization [63]. Nevertheless, Flyvbjerg [63] contested that one case study can be generalized and has been used by old scientists like Galileo, Newton, Einstein, and Bohr. A single case study generalization can be done with careful design and implementation. It is still a matter of selecting and designing a single case study, and having a huge number of case studies does not guarantee greater generalization. This study presents a design for the use of case studies in penetration testing model evaluation, besides determining the number of case studies and testers to be used during this evaluation in order to improve the generalization of the case study results.

Deciding the number of case studies is a difficult task, as there is no limit to the maximum number of case studies [37]. Table 3 summarizes our reviews to support the use of a single test case when conducting an evaluation. These studies, published between 2004–2015, support the use of a single case study as it helps researchers focus on the main components of their research and reduce the complexity and time required for evaluation.

Table 3. Single case study in literature

Reference	Criteria	Single case study characteristics
[64]	Credibility	It represents a more important value than having large-scale cases with less credible results.
[8]	Complexity	It reduces complexity.
[60]		It reduces complexity and supports unique cases.
[87]	Uniqueness	It details an examination that investigates a single subject or model in a specific, unique, bounded system.
[78]		The choice for evaluation when the case study is a unique case, at least for the researcher of the study.
[63]	Generalization	It can be generalized.
[37]		There is no limit to the maximum number of case studies meanwhile many times one is enough.
[68]		It supports and demonstrates a model or theory. Single-case studies are strongest in describing the results and findings.
[62]		In the case study, the data will be collected from the same project development which makes a single case study enough.
[70]		It is limited to other cases that have the same characteristics. Improve the research data describing the case study.
[58]		Many software engineering uses a single case study.
[61]	Reusability	It provides reusability that can be used by other researchers and the reduce time.
[79]	Flexibility	It provides more flexibility and exploratory than having multiple case studies.
[77]	In-depth	It provides in-depth detail of the results and findings.
[71]		It provides a deeper understanding of the important issues. It allows the researchers to observe, explore, and explain the results and findings in real-life environment variables.
[80]		It helps in finding the correlations and prioritization the results.
[72]		It should be employed, mainly when researching a previously unsearched domain. It makes the results to be investigated in depth.

Ghauri [37], Yin [8], and Flyvbjerg [63] have discussed the number of case studies, and they all agreed on the difficulty of selecting the number of case studies. At the same time, they all agree that one case study is enough for generalization, especially if we deal with highly complex implementations. Using a single case study may help in making the case study generalized and centralized to scientific development, which enriches the research with a deeper focus on the important components rather than replicating the same process multiple times. A single case study is an empirical inquiry and a detailed examination that investigates a single subject or model in a specific, unique, and bounded system [87]. Perry et al. [60] and McLeod and Elliott [64] agreed that in evaluating models and theories, it is sufficient to use a single case study. A single case study can be helpful to represent complex and unique cases, while the result represents a more important value than large-scale cases with less credible results. Also, Zivkovic [68] found that a single case can be used to support and demonstrate a model or theory and is strongest in describing the results and findings.

A single case study can reduce time while enhancing results. It can also support reusability, which can be used by other researchers [61]. Similarly, Seaman [62] found that in the single case study, the data will be collected from the same project, which makes it sufficient. Moreover, Wong et al. [70] found that the observation made on a single case

study is limited to other cases that have the same characteristics, which in fact helps to present the limitation, which represents the scope of the studies. Therefore, researchers need a detailed description of the characteristics of the case study and the evaluation limitations that make a single case study a better choice.

In software engineering, such as testing, projects use a single case study [14, 58, 59]. Stol and Fitzgerald [77] agreed on the fact that a single case study can be used to provide in-depth details of the results and findings. While Hanssen [78] found that a single case study is a good choice for evaluation, especially when the case study is a unique case, at least for the researcher of the study. Da Silva et al. (2014) also found that a single case study can provide more flexibility and explanatory power than multiple case studies. Also, Zogaj et al. [71] stated that single case studies provide a deeper understanding of the important issues when the data are limited, as it allows the researchers to observe, explore, and explain the results and findings in real-life environmental variables. Likewise, Sánchez et al. [80] found that a single case study helps in finding correlations and prioritizing the results. Similarly, Koskosas & Koskosa [72] mentioned that a single case study should be used when studying a previously unexplored domain because it allows the findings to be thoroughly analyzed.

On the other hand, some researchers showed that using a single case study may narrow the scope of generalization, while multiple case studies may increase it [61]. We found in the literature that a previous penetration testing thesis also used two case studies with one tester [72, 74, 81, 117–120]. More than two test cases were very rare in the penetration testing domain. Authors have extensively reviewed the publications from the selected online databases and hardly found model evaluations that used more than two case studies. For example, three test cases with one tester were used by Goseva-Popstojanova & Perhinschi [121], Al-Azzani et al. [122], and Mouelhi et al. [123]. Therefore, to keep the evaluation framework and its results clear and readable, improve the reliability and generalization of the model evaluation framework, and reduce the criticisms of the single case study, we suggest using two case studies. The number of case studies required is strongly linked to the complexity of the application under test. It is argued that a two-case study approach is required to cater to multiple types of applications, i.e., mobile, desktop, cloud, and hybrid implementations like mobile cloud computing and fog computing, where it can be used for static and dynamic offloading [29–31, 51, 52].

To conclude, in the context of penetration testing, a single case study is sufficient to generalize the research and may enable the research to determine and identify the evaluation results and findings, which are consistent with what was found and practiced by Ahmad et al. [124], Pandey & Mishra [125], and Ceric & Holland [126]. However, using two case studies is thus sufficient for the evaluation and still identifiable by presenting the details of each case study, as well as protecting the study from single case study criticisms. Using two case studies was also practiced previously in multiple penetration testing studies such as [118, 127, 128]. Therefore, the proposed design suggests that one may use two case studies to provide deep, in-depth results within a reasonable amount of time as well as protect the research from single-case study criticism.

3-3- Number of Testers

As implied in the name, single-case designs have traditionally involved the use of a single tester [65, 69, 86]. In this proposed design, the participant is referred to as the tester. A single case study is a comprehensive analysis of a single collection of data, a single subject or item, or a single depository in a single case, rather than the methods of investigation used [87]. It has been defined as an individual "case" unit of intervention and unit of data analysis that may include a single participant [67].

This section will review the literature of previous penetration testing model evaluations in order to answer RQ4 (How many test cases should be used to evaluate the penetration testing model?). By the end of this section, a discussion is provided to conclude and recommend the number of case studies to be used in evaluating penetration testing, which answers RQ5 (How many testers will be needed to assess the penetration testing model?).

The single tester evaluation is applied in this proposed design as it is an accepted evaluation practice in penetration testing as it was practiced previously in multiple studies such as Hanna [82], Doupé [84], Pan [83], Sánchez et al. [80], and Zhou [81]. In particular, it is also a recognized evaluation standard in penetration testing related to security testing, software security, and software testing. Moreover, the single tester/participant evaluation has been successfully applied in software engineering, as illustrated in Figure 3.

Figure 3 depicts the software engineering domain and sub-domains of a single tester or participant and a single case study evaluation. There are five studies in penetration testing that have applied single tester evaluation [80–84], two under security testing [70, 71], three under software security [72, 73, 129], three under software testing [74–76], and five under software engineering [60, 61, 77–79]. Since 1995 (Wong et al. [70]) and as recently as 2015 [80], single tester evaluation has been an accepted standard in these related areas, especially penetration testing.

Among the key rationale why single tester evaluation is recognized in penetration testing, domains are the enormous input data size and the huge corresponding results. There is a need to analyze this huge volume of results efficiently and present them clearly in order to draw conclusions and justify findings. Therefore, one tester is sufficient for evaluation purposes, as it allows the researcher to explore the relationship between a phenomenon and its context in greater detail, and thus uncover important context variables that may otherwise be missed [130].

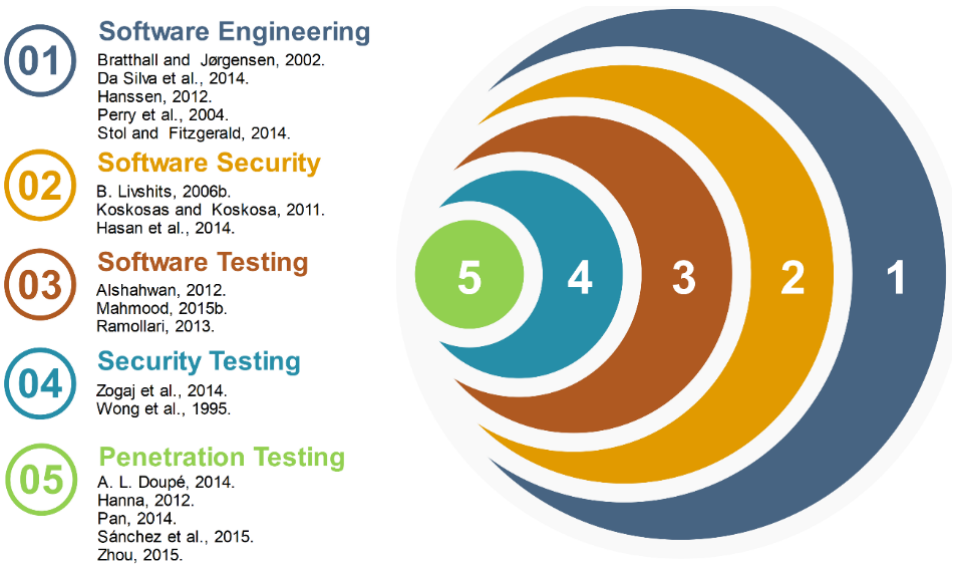

Figure 3. Software engineering domain and sub-domains single tester/participant and single case study evaluation

Using a single tester to evaluate models is a common practice that has been used in many studies that have been published in reputable journals. Table 4 lists many studies that have used a single tester to evaluate their model in multiple domains. In this context, we have summarized the results of reviewing certain Ph.D. and master's theses from well-reputable universities that use a single participant, which is equivalent to a single tester in penetration testing, in Table 5. A single case study helps researchers manage and enhance the quality of their research. Table 6 summarizes the main characteristics of using a single tester that can support the use of one tester in evaluating the penetration testing model.

Table 4. List of studied used one single tester in model evaluation

Reference	Domain	Title
Livshits [73]	Software security	Improving software security.
Zhou [81]		Improving security and privacy of integrated web applications.
Livshits & Lam [131]		Finding security vulnerabilities in java applications with static analysis.
Mahmood [75]		An evolutionary approach for system testing of android applications.
Pan [83]		Cybersecurity testing and intrusion detection for cyber-physical power systems.
Doupé [84]	Penetration testing	Advanced automated web application.
Hanna [82]		A hybrid framework for the systematic detection of software security vulnerabilities in source code.
Al-Azzani [122]		Architecture-centric testing for security.
Alshahwan [76]	Software testing	Utilizing output in web application server-side testing.
Ramollari [74]		Automated runtime testing of web services.

Table 5. List of PhD and Master thesis used one single in model evaluation

	Title	University
1	Improving software security with precise static and runtime analysis [73]	Stanford University
2	Improving security and privacy of integrated Web applications [81]	University of Virginia
3	Automated runtime testing of Web services [74]	University of Sheffield
4	A hybrid framework for the systematic detection of software security vulnerabilities in source code [82]	Concordia University
5	An evolutionary approach for system testing of android applications [75]	George Mason University
6	Cybersecurity testing and intrusion detection for cyber-physical power systems [83]	Mississippi State University
7	Advanced automated Web application vulnerability analysis [84]	University Of California
8	Utilizing output in Web application server-side testing [76]	University College London

Table 6. Single tester characteristics

Reference	Title	Single tester characteristics
[85]	Rey: An intensive single case study of a probation youth with immigrant background	Allow for sensitization. Increase awareness for the results.
[86]	A case study of extensive reading with an unmotivated L2 reader	Provide in-depth analysis.
[66]	Types of case study work: A conceptual framework for case-based research	Enhance resources efficiency.
[69]	Single-case research	Improve controlling the variables.

Based on the above discussion, it can be argued that a single participant is acceptable in the academic domain for researchers and students. It shows that many studies that have been published in a scientific journal have used single participants. Furthermore, several Ph.D. and master's theses have been conducted using a single-participant approach. Thus, a single participant can be used in evaluating the penetration testing model, as it is a generally accepted approach in the academic domain, especially if the complexity of the evaluation of the penetration testing model is considered.

The single participant allows for sensitization and increases awareness of the results [85]. Furthermore, the single participant helps to make the primary analysis of a single individual with specific characteristics [86]. Edwards [66] found that resources are more likely to be available to obtain information from multiple sources when using a single participant. Similarly, Barker et al. [69] found that one participant is valuable and helps in controlling the variables in order to get meaningful results that detect positive and negative points.

3-4- Select Testbed Applications

Testbed applications are a tool to be used in evaluating a model. In the context of penetration testing, a testbed application represents a vulnerable application that has some known vulnerabilities [132, 133]. This application will be used as an input to the testing process, where the evaluation will be done between the number of vulnerabilities exposed by the model versus the real number of vulnerabilities [134]. There are many examples of applications that can be used as testbed applications, such as GoatDroid, OWASP web test application, HerdFinancial, and FourGoats [135, 136].

Good testbed applications must be reliable, compatible, and extensible. Reliable means it should be evaluated before being used in evaluation [137, 138]. While compatibility means that the application must be visible to be used as an application within the domain understudy [139]. Similarly, the application must be extensible to add new features and functionality [140, 141]. These requirements should be mapped and applied in each step of the selection of the testbed application for the evaluation of any proposed penetration testing model.

3-5- Environment Preparation

The application intended to be tested under penetration testing needs to run in order to execute the prepared test cases [142]. The process of converting a testbed application into a running application requires deploying the testbed application's required services [143]. Therefore, environment preparation must include two phases: backend and installation [144, 145]. The first phase prepares the backend part of the environment. This includes the selection of the service providers to use and the deployment of these services. The second phase of environmental preparation includes the selection of devices and the installation process.

Each case study needs certain parameters and instances to be prepared before conducting penetration testing in order to emulate the real-life scenario [142]. For example, gateway addresses and service addresses need to be configured, as well as the library's location, DNS address, and firewall setup attributes. These preparations must be specifically implemented and structured in order to assist the researcher in presenting the method and findings more effectively [145, 146]. Since many of the environmental variables are shared through multiple case studies, having solid environmental preparation aids in implementing the other case studies.

4- Discussion

4-1- Evaluation Using Case Study

As shown in the previous sections. The process of case study design for penetration testing model evaluation follows certain steps: (i) Select the number of case studies. (ii) Select the number of testers. (iii) Select testbed applications for the case studies. (iv) prepares the application environment. This process represents the answer to RQ1 (what are the steps to design penetration testing model evaluation using a case study?). The following discussion will conclude the previous penetration testing models' number of test cases and testers used and present the recommended numbers of test cases and testers to be used when evaluating penetration testing models that answer research questions 2–5.

RQ2: *What is the number of test cases used in the related research?*

In the area of software engineering, a single case study with one participant is sufficient for generalization [147]. Other fields accept that a single case study with one sample is sufficient for generalization [37, 63]. In most single-case research, selection is generally not a concern, even if one participant is exposed [67]. The single case study with a single participant has been used previously as it simplifies the model evaluation framework [148].

RQ3: *What is the number of testers used in the related research?*

Although some penetration and security testing and evaluation frameworks have used a single tester for one case study [74, 76, 82], others have used two case studies [73]. A single case study with one tester has been applied to focus on the details of the case study [74, 76, 82], while two case studies with a single tester have been used to efficiently determine the effect of their proposed manifests [73]. Others have also used a single case study with one tester to focus on the details of the case study [74, 76, 82].

RQ4: *How many test cases should be used to evaluate the penetration testing model?*

RQ5: *How many testers will be needed to assess the penetration testing model?*

The proposed design of the evaluation will be based on two case studies: one will be used for static and the other for dynamic offloading, with one penetration tester. Using a single case study with a single participant is sufficient; however, using two case studies with one tester will improve evaluation generalization and make the cases more identifiable by presenting the details of each case study. Using one tester's main advantages are: (i) accepted in the areas of penetration testing, security testing, software security, software testing, and software engineering; (ii) sufficient for generalization; (ii) allows the researcher to explore the relationship between a phenomenon and its context.

4-2- Main Findings of the Present Study

The findings of this paper lead to the conclusion that case study design for penetration testing model evaluation has to start with determining the number of case studies and number of testers, then selecting and preparing testbed applications. Those steps are critical and subject to criticism, which requires researchers to support his decision with solid literature and logic. This paper has discussed many studies that evaluated penetration and security testing models and frameworks using a single tester for one case study and other studies that used two case studies. This paper, by studying and analyzing previous works, has found that designing the testing models and frameworks for evaluation based on two case studies with one tester is sufficient. The findings also revealed that two case studies with one tester will provide the researchers with additional data, protect them from the criticism of the single case study, and improve the generalization and clarity of the results.

4-3- Comparison with Other Studies

In this paper, we reviewed studies in a variety of domains, including penetration testing, security testing, software security, software testing, and software engineering, and analyzed their case study evaluation method framework to derive framework guidelines that can be used to construct a solid, efficient, and generalizable structure for a penetration testing evaluation case study. The proposed case study design in this paper was adopted from the previously proposed [7, 12, 14, 18, 38, 58, 114–116] case study design models, frameworks, methods, and guidelines provided. Furthermore, this paper has found that using a single test case when conducting a sufficient evaluation was previously used by Ahmad et al. [124], Pandey and Mishra [125], and Ceric and Holland [126], while two case studies are more identifiable and can support generalization more, which were used in multiple penetration testing studies such as Chung, Mueller, and Kim [118, 127, 128]. In the same context, a single participant is sufficient, which is also supported by other studies (e.g., [66, 69, 85, 86]).

4-4- Implication and Explanation of Findings

Simplifying the process by utilizing a single case study for model evaluation, specifically in the domain of penetration testing, will encourage researchers and practitioners to become more involved in the research domain by reducing complexity and time. Using a single case study can also be very beneficial in applying the models, testing tools, and guidelines while also improving the quality of the results and safeguarding the model from generalization criticism by using a well-structured and solid approach.

There have been few studies to standardize the usage of the case study approach in the domain of software engineering and even fewer in the domain of software testing. Future directions should be toward developing rules and standards that provide metrics for assessing the evaluation process while meeting the expectations of testers, researchers, and practitioners about the use of case studies in the domain of software testing. These metrics should take into account resource constraints as well as the need for increased production to meet the fast-evolving technologies in this domain in order to improve the quality of information systems globally in terms of security, performance, usability, accessibility, and functionality.

5- Conclusion

This work provided an outline of the dilemma that many researchers confront when deciding on the number of case studies and testers to use in the realm of penetration testing. In this paper, we offer an adapted framework for designing a case study for penetration testing evaluation. Certain decisions in the suggested design were made based on published research in multiple fields, particularly software engineering. We highlighted current practices, as well as the pros and cons of each choice, and then offered the option that met the penetration testing model evaluation. This paper also proposes case study design steps that summarize the main phases of implementing case study evaluation for penetration testing models. The proposed case study design steps will guide the researchers in terms of the main processes and main decisions that they will encounter during the evaluation process of their penetration testing utilizing the case study approach. This paper presented the challenge of using single or multiple case studies and testers to evaluate penetration testing, as well as the current practices and the criticism they encounter.

Moreover, this study revealed that using a single case study with a single tester is sufficient; however, using two case studies with one tester is a generally accepted practice because it improves generalization, improves case identification, and assists researchers in exploring the relationship between a phenomenon and its context. This study may help academics and practitioners evaluate their methodologies, particularly in determining the number of case studies and testers.

6- Declarations

6-1- Author Contributions

Conceptualization, A.A. and H.K.; methodology, A.A., H.K., and Y.Z.; investigation, A.A. and H.K.; resources, A.A., H.K., and Y.Z.; data curation, A.A. and H.K.; writing—original draft preparation, A.A., H.K., and Y.Z.; writing—review and editing, A.A., H.K., and Y.Z.; visualization, H.K. All authors have read and agreed to the published version of the manuscript.

6-2- Data Availability Statement

Data sharing is not applicable to this article.

6-3- Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6-4- Institutional Review Board Statement

Not applicable.

6-5- Informed Consent Statement

Not applicable.

6-6- Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript. In addition, the ethical issues, including plagiarism, informed consent, misconduct, data fabrication and/or falsification, double publication and/or submission, and redundancies have been completely observed by the authors.

7- References

[1] Godin, G. (1994). Social-cognitive models. Advances in exercise adherence, 113–136, Human Kinetics Publishers, Champaign, United States.

[2] Keng Siau, & Rossi, M. (1998). Evaluation of information modeling methods-a review. Proceedings of the Thirty-First Hawaii International Conference on System Sciences. doi:10.1109/hicss.1998.648327.

[3] Yuan, Q., Pi, Y., Kou, L., Zhang, F., Li, Y., & Zhang, Z. (2022). Multi-Source Data Processing and Fusion Method for Power Distribution Internet of Things Based on Edge Intelligence. Frontiers in Energy Research, 10. doi:10.3389/fenrg.2022.891867.

[4] Sitthimongkolchai, N., Viriyavejakul, C., & Tuntiwongwanich, S. (2022). The BEE Model with Live Virtual Classroom to Enhancing Creative Works. Emerging Science Journal, 6, 108–122. doi:10.28991/esj-2022-sied-08.

[5] Shiffrin, R. M., Lee, M. D., Kim, W., & Wagenmakers, E. J. (2008). A survey of model evaluation approaches with a tutorial on hierarchical bayesian methods. Cognitive Science, 32(8), 1248–1284. doi:10.1080/03640210802414826.

[6] Gao, K., Wang, Z., Mockus, A., & Zhou, M. (2023). On the Variability of Software Engineering Needs for Deep Learning: Stages, Trends, and Application Types. IEEE Transactions on Software Engineering, 49(2), 760–776. doi:10.1109/TSE.2022.3163576.

[7] Hammersley, M.; Foster, P. and Gomm, R. (2000). Case study and generalization. Case Study Method: Key Issues, Key Texts, 98-115. Sage, London, United Kingdom.

[8] Yin, R. K. (2013). Validity and generalization in future case study evaluations. Evaluation, 19(3), 321–332. doi:10.1177/1356389013497081.

[9] Wieringa, R., & Daneva, M. (2015). Six strategies for generalizing software engineering theories. Science of Computer Programming, 101, 136–152. doi:10.1016/j.scico.2014.11.013.

[10] Alfayez, R., Ding, Y., Winn, R., & Alfayez, G. (2022). What is Discussed About Software Engineering Ethics on Stack Exchange (Q&A) Websites? A Case Study. 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA). doi:10.1109/sera54885.2022.9806760.

[11] Deshmukh, S. A., & Kasar, S. L. (2022). Significance of Software Engineering Phases in the Development of a Software Application. Designing User Interfaces with a Data Science Approach, 111–132, IGI Global, Hershey, United States. doi:10.4018/978-1-7998-9121-5.ch006.

[12] Aha, D. W. (1992). Generalizing from Case Studies: A Case Study. Machine Learning Proceedings 1992, 1–10, Morgan Kaufmann, Burlington, United States. doi:10.1016/b978-1-55860-247-2.50006-1.

[13] Bosua, R., Cheong, M., Clark, K., Clifford, D., Coghlan, S., Culnane, C., Leins, K., & Richardson, M. (2022). Using public data to measure diversity in computer science research communities: A critical data governance perspective. Computer Law & Security Review, 44, 105655. doi:10.1016/j.clsr.2022.105655.

[14] Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131–164. doi:10.1007/s10664-008-9102-8.

[15] Klotins, E., Gorschek, T., Sundelin, K., & Falk, E. (2022). Towards cost-benefit evaluation for continuous software engineering activities. Empirical Software Engineering, 27(6). doi:10.1007/s10664-022-10191-w.

[16] Dakkak, A., Bosch, J., & Olsson, H. H. (2022). Controlled Continuous Deployment: A Case Study From The Telecommunications Domain. Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering. doi:10.1145/3529320.3529323.

[17] Andersson, C., & Runeson, P. (2007). A spiral process model for case studies on software quality monitoring - Method and metrics. Software Process Improvement and Practice, 12(2), 125–140. doi:10.1002/spip.311.

[18] Mizuno, O., Ikami, S., Nakaichi, S., & Kikuno, T. (2007). Fault-Prone Filtering: Detection of Fault-Prone Modules Using Spam Filtering Technique. First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007) Madrid, Spain. doi:10.1109/esem.2007.29.

[19] Özakıncı, R., & Kolukısa Tarhan, A. (2023). A decision analysis approach for selecting software defect prediction method in the early phases. Software Quality Journal, 31(1), 121–177. doi:10.1007/s11219-022-09595-0.

[20] Sangal, N., Jordan, E., Sinha, V., & Jackson, D. (2005). Using dependency models to manage complex software architecture. ACM SIGPLAN Notices, 40(10), 167–176. doi:10.1145/1103845.1094824.

[21] Sobhy, D., Minku, L., Bahsoon, R., & Kazman, R. (2022). Continuous and Proactive Software Architecture Evaluation: An IoT Case. ACM Transactions on Software Engineering and Methodology, 31(3), 1–54. doi:10.1145/3492762.

[22] Geer, D., & Harthorne, J. (2002). Penetration testing: a duet. 18th Annual Computer Security Applications Conference, 2002. Proceedings. doi:10.1109/csac.2002.1176290.

[23] Goel, J. N., Asghar, M. H., Kumar, V., & Pandey, S. K. (2016). Ensemble based approach to increase vulnerability assessment and penetration testing accuracy. 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH). doi:10.1109/iciccs.2016.7542303.

[24] Xiong, P., & Peyton, L. (2010). A model-driven penetration test framework for Web applications. 2010 Eighth International Conference on Privacy, Security and Trust. doi:10.1109/pst.2010.5593250.

[25] Xu, W., Groves, B., & Kwok, W. (2016). Penetration testing on cloud---case study with own cloud. Global Journal of Information Technology, 5(2), 87. doi:10.18844/gjit.v5i2.198.

[26] Zhao, J., Shang, W., Wan, M., & Zeng, P. (2015). Penetration testing automation assessment method based on rule tree. 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER). doi:10.1109/cyber.2015.7288225.

[27] Meyers, B. S., Almassari, S. F., Keller, B. N., & Meneely, A. (2022). Examining Penetration Tester Behavior in the Collegiate Penetration Testing Competition. ACM Transactions on Software Engineering and Methodology, 31(3), 1–25. doi:10.1145/3514040.

[28] Barik, K., Konar, K., Banerjee, A., Das, S., Abirami, A. (2022). An Exploration of Attack Patterns and Protection Approaches Using Penetration Testing. Intelligent Data Communication Technologies and Internet of Things. Lecture Notes on Data Engineering and Communications Technologies, Volume 101. Springer, Singapore. doi:10.1007/978-981-16-7610-9_36.

[29] Al-Ahmad, A. S., Aljunid, S. A., & Ismail, N. K. (2020). Mobile cloud computing applications penetration testing model design. International Journal of Information and Computer Security, 13(2), 210–226. doi:10.1504/IJICS.2020.108849.

[30] Al-Ahmad, A. S., Aljunid, S. A., & Sani, A. S. A. (2013). Mobile Cloud Computing Testing Review. 2013 International Conference on Advanced Computer Science Applications and Technologies. doi:10.1109/acsat.2013.42.

[31] Al-Ahmad, A. S., Kahtan, H., Hujainah, F., & Jalab, H. A. (2019). Systematic Literature Review on Penetration Testing for Mobile Cloud Computing Applications. IEEE Access, 7, 173524–173540. doi:10.1109/ACCESS.2019.2956770.

[32] Ali, A., Ahmed, M., Imran, M., & Khattak, H. A. (2020). Security and Privacy Issues in Fog Computing. Fog Computing, 105–137, John Wiley & Sons, Hoboken, United States. doi:10.1002/9781119551713.ch5.

[33] Alzoubi, Y. I., Al-Ahmad, A., Jaradat, A., & Osmanaj, V. H. (2021). FOG computing architecture, benefits, security, and privacy, for the internet of thing applications: An overview. Journal of Theoretical and Applied Information Technology, 99(2), 436–451.

[34] Alzoubi, Y. I., Al-Ahmad, A., & Jaradat, A. (2021). Fog computing security and privacy issues, open challenges, and blockchain solution: An overview. International Journal of Electrical and Computer Engineering (IJECE), 11(6), 5081. doi:10.11591/ijece.v11i6.pp5081-5088.

[35] Alzoubi, Y. I., Osmanaj, V. H., Jaradat, A., & Al‐Ahmad, A. (2020). Fog computing security and privacy for the Internet of Thing applications: State-of-the-art. Security and Privacy, 4(2). doi:10.1002/spy2.145.

[36] Maray, M., & Shuja, J. (2022). Computation Offloading in Mobile Cloud Computing and Mobile Edge Computing: Survey, Taxonomy, and Open Issues. Mobile Information Systems, 2022. doi:10.1155/2022/1121822.

[37] Ghauri, P. (2004). Designing and conducting case studies in international business research. In Handbook of Qualitative Research Methods for International Business, 109–124, Edward Elgar, Cheltenham, United Kingdom. doi:10.4337/9781781954331.00019.

[38] Gehrke, S., Niemz, S., Wenige, L., & Ruhland, J. (2022). Investigation of Senior IT Management Skills Using COBIT Enablers and Social Media Platform. Journal of Human, Earth, and Future, 3(1), 69-81. doi:10.28991/HEF-2022-03-01-05.

[39] Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. Education for Information, 22(2), 63–75. doi:10.3233/EFI-2004-22201.

[40] Engebretson, P. (2013). The basics of hacking and penetration testing: ethical hacking and penetration testing made easy, Elsevier, Amsterdam, Netherlands. doi:10.1016/C2013-0-00019-9.

[41] Arkin, B., Stender, S., & McGraw, G. (2005). Software penetration testing. IEEE Security and Privacy, 3(1), 84–87. doi:10.1109/MSP.2005.23.

[42] Whitaker, A., & Newman, D. P. (2005). Penetration Testing and Network Defense: Penetration Testing _1. Cisco Press, Indianapolis, United States.

[43] Yeo, J. (2013). Using penetration testing to enhance your company's security. Computer Fraud & Security, 2013(4), 17-20. doi:10.1016/S1361-3723(13)70039-3.

[44] Sanjaya, I. G. A. S., Sasmita, G. M. A., & Sri Arsa, D. M. (2020). Information technology risk management using ISO 31000 based on issaf framework penetration testing (Case study: Election commission of x city). International Journal of Computer Network and Information Security, 12(4), 30–40. doi:10.5815/ijcnis.2020.04.03.

[45] Stepanova, T., Pechenkin, A., & Lavrova, D. (2015). Ontology-based big data approach to automated penetration testing of large-scale heterogeneous systems. Proceedings of the 8th International Conference on Security of Information and Networks. doi:10.1145/2799979.2799995.

[46] Arjun, C. V. (2019). Penetration Testing for Denial of Service Attacks and its Variants. Journal of Advanced Research in Dynamical and Control Systems, 10, 2320-2326.

[47] Arnaldy, D., & Perdana, A. R. (2019). Implementation and Analysis of Penetration Techniques Using the Man-In-The-Middle Attack. 2019 2nd International Conference of Computer and Informatics Engineering (IC2IE), Banyuwangi, Indonesia. doi:10.1109/ic2ie47452.2019.8940872.

[48] Wibowo, R. M., & Sulaksono, A. (2021). Web Vulnerability Through Cross Site Scripting (XSS) Detection with OWASP Security Shepherd. Indonesian Journal of Information Systems, 3, 149–159. doi:10.24002/ijis.v3i2.4192.

[49] Al-Ahmad, A., Ata, B. A., & Wahbeh, A. H. S. (2012). Pen testing for Web applications. International Journal of Information Technology and Web Engineering, 7(3), 1–13. doi:10.4018/jitwe.2012070101.

[50] Salis, A. (2021). Towards the Internet of Behaviors in Smart Cities through a Fog-To-Cloud Approach. HighTech and Innovation Journal, 2(4), 273-284. doi:10.28991/HIJ-2021-02-04-01.

[51] Al-Ahmad, A. S., & Kahtan, H. (2018). Test case selection for penetration testing in mobile cloud computing applications: A proposed technique. Journal of Theoretical and Applied Information Technology, 96(13), 4238–4248.

[52] Al-Ahmad, A.S., Kahtan, H. (2019). Fuzz Test Case Generation for Penetration Testing in Mobile Cloud Computing Applications. Intelligent Computing & Optimization. ICO 2018, Advances in Intelligent Systems and Computing, 866, Springer, Cham, Switzerland. doi:10.1007/978-3-030-00979-3_27.

[53] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). doi:10.1109/icse-seip.2019.00042.

[54] Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004). Evidence-based software engineering. Proceedings. 26th International Conference on Software Engineering. doi:10.1109/icse.2004.1317449.

[55] Kitchenham, B., Pickard, L., & Pfleeger, S. L. (1995). Case Studies for Method and Tool Evaluation. IEEE Software, 12(4), 52–62. doi:10.1109/52.391832.

[56] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering version 2.3. Engineering, 45(4ve), 1051.

[57] Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software, 80(4), 571–583. doi:10.1016/j.jss.2006.07.009.

[58] Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). Scaling up Case Study Research to Real‐World Software Practice. Case Study Research in Software Engineering; Guidelines and Examples, 97-107, John Wiley & Sons, Hoboken, United States. doi:10.1002/9781118181034.ch7.

[59] Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, Hoboken, united States. doi:10.1002/9781118181034.

[60] Perry, D. E., Sim, S. E., & Easterbrook, S. M. (2004). Case studies for software engineers. Proceedings. 26th International Conference on Software Engineering. doi:10.1109/icse.2004.1317512.

[61] Bratthall, L., & Jørgensen, M. (2002). Can you trust a single data source exploratory software engineering case study? Empirical Software Engineering, 7(1), 9–26. doi:10.1023/A:1014866909191.

[62] Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering, 25(4), 557–572. doi:10.1109/32.799955.

[63] Flyvbjerg, B. (2011). Five Misunderstandings About Case-Study Research. Qualitative Research Practice, 12, 390–404. doi:10.4135/9781848608191.d33.

[64] McLeod, J., & Elliott, R. (2011). Systematic case study research: A practice-oriented introduction to building an evidence base for counselling and psychotherapy. Counselling and Psychotherapy Research, 11(1), 1–10. doi:10.1080/14733145.2011.548954.

[65] Kratochwill, T. R., & Levin, J. R. (2010). Enhancing the Scientific Credibility of Single-Case Intervention Research: Randomization to the Rescue. Psychological Methods, 15(2), 124–144. doi:10.1037/a0017736.

[66] Edwards, D. J. A. (1998). Types of case study work: A conceptual framework for case-based research. Journal of Humanistic Psychology, 38(3), 36–70. doi:10.1177/00221678980383003.

[67] Kratochwill, T. R., Hitchcock, J., Horner, R. H., Levin, J. R., Odom, S. L., Rindskopf, D. M., & Shadish, W. R. (2010). Single-case designs technical documentation. What works clearinghouse. Institute of Education Sciences. Available online: http://ies.ed.gov/ncee/wwc/pdf/wwc_scd.pdf (accessed on April 2023).

[68] Zivkovic, J. (2012). Strengths and Weaknesses of Business Research Methodologies: Two Disparate Case Studies. Business Studies Journal, 4(2), 91-99.

[69] Barker, J., McCarthy, P., Jones, M., & Moran, A. (2011). Single case research methods in sport and exercise. Routledge, London, United Kingdom. doi:10.4324/9780203861882.

[70] Wong, W. E., Horgan, J. R., London, S., & Mathur, A. P. (1995). Effect of test set minimization on fault detection effectiveness. Proceedings of the 17th International Conference on Software Engineering- ICSE '95. doi:10.1145/225014.225018.

[71] Zogaj, S., Bretschneider, U., & Leimeister, J. M. (2014). Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary. Journal of Business Economics, 84(3), 375–405. doi:10.1007/s11573-014-0721-9.

[72] Koskosas, I., & Koskosa, M. M. (2011). Internet banking security management through trust management. World Computer Science and Information Technology Journal (WCSIT), 1(3), 79-87.

[73] Livshits, B. (2006). Improving software security with precise static and runtime analysis. Ph.D. Thesis, Stanford University, Stanford, United States.

[74] Ramollari, E. (2013) Automated Runtime Testing of Web Services. Ph.D. Thesis, University of Sheffield, Sheffield, United Kingdom.

[75] Mahmood, R. (2015). An evolutionary approach for system testing of android applications. Ph.D. Thesis, George Mason University, Fairfax, United States.

[76] Alshahwan, N; (2012) Utilizing Output in Web Application Server-Side Testing. Ph.D. Thesis, University College London (UCL), London, United Kingdom.

[77] Stol, K. J., & Fitzgerald, B. (2014). Two's company, three's a crowd: a case study of crowdsourcing software development. Proceedings of the 36th International Conference on Software Engineering. doi:10.1145/2568225.2568249.

[78] Hanssen, G. K. (2012). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. Journal of Systems and Software, 85(7), 1455–1466. doi:10.1016/j.jss.2011.04.020.

[79] Da Silva, I. F., Da Mota Silveira Neto, P. A., O'Leary, P., De Almeida, E. S., & Meira, S. R. D. L. (2014). Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study. Journal of Systems and Software, 88(1), 189–206. doi:10.1016/j.jss.2013.10.040.

[80] Sánchez, A. B., Segura, S., Parejo, J. A., & Ruiz-Cortés, A. (2017). Variability testing in the wild: the Drupal case study. Software and Systems Modeling, 16(1), 173–194. doi:10.1007/s10270-015-0459-z.

[81] Zhou, Y. (2015). Improving Security and Privacy of Integrated Web Applications. Ph.D. Thesis, University of Virginia, Charlottesville, United States.

[82] Hanna, Aiman (2012) A Hybrid Framework for the Systematic Detection of Software Security Vulnerabilities in Source Code. PhD Thesis, Concordia University, Montreal, Canada.

[83] Pan, S. (2014). Cybersecurity testing and intrusion detection for cyber-physical power systems. Ph.D. Thesis, Mississippi State University, Starkville, United States.

[84] Doupé, A. L. (2014). Advanced automated web application vulnerability analysis. Ph.D. Thesis, University of California, Oakland, United States.

[85] Lutz, B. (2013). Rey: An intensive single case study of a probation youth with immigrant background participating in wraparound Santa Cruz. PhD Thesis, The Chicago School of Professional Psychology, Chicago, United States.

[86] Ro, E. (2013). A case study of extensive reading with an unmotivated L2 reader. Reading in a Foreign Language, 25(2), 213-233.

[87] Bonilla, P. (2015). Is our food safe? An Assessment: on the European Union food safety policy, concerning the safety of meat & animal-derived food products in the EU, Master Thesis, University of Twente, Enschede, Netherlands.

[88] Bastian, H., Glasziou, P., & Chalmers, I. (2010). Seventy-five trials and eleven systematic reviews a day: How will we ever keep up? PLoS Medicine, 7(9), 1000326. doi:10.1371/journal.pmed.1000326.

[89] Sana, M. U., & Li, Z. (2021). Efficiency aware scheduling techniques in cloud computing: A descriptive literature review. PeerJ Computer Science, 7, 1–37. doi:10.7717/PEERJ-CS.509.

[90] Ali, O., Shrestha, A., Jaradat, A., & Al-Ahmad, A. (2022). An Evaluation of Key Adoption Factors towards Using the Fog Technology. Big Data and Cognitive Computing, 6(3), 81. doi:10.3390/bdcc6030081.

[91] Alzoubi, Y. I., Al-Ahmad, A., Kahtan, H., & Jaradat, A. (2022). Internet of Things and Blockchain Integration: Security, Privacy, Technical, and Design Challenges. Future Internet, 14(7), 216. doi:10.3390/fi14070216.

[92] AlAhmad, A. S., Kahtan, H., Alzoubi, Y. I., Ali, O., & Jaradat, A. (2021). Mobile cloud computing models security issues: A systematic review. Journal of Network and Computer Applications, 190, 103152. doi:10.1016/j.jnca.2021.103152.

[93] Alzoubi, Y. I., Gill, A. Q., & Al-Ani, A. (2016). Empirical studies of geographically distributed agile development communication challenges: A systematic review. Information and Management, 53(1), 22–37. doi:10.1016/j.im.2015.08.003.

[94] Evans, D. (2003). Hierarchy of evidence: A framework for ranking evidence evaluating healthcare interventions. Journal of Clinical Nursing, 12(1), 77–84. doi:10.1046/j.1365-2702.2003.00662.x.

[95] Kahtan, H., Bakar, N. A., & Nordin, R. (2012, October). Reviewing the challenges of security features in component based software development models. 2012 IEEE Symposium on E-Learning, E-Management and E-Services, Kuala Lumpur, Malaysia. doi:10.1109/IS3e.2012.6414955.

[96] Kahtan, H., Bakar, N. A., & Nordin, R. (2014). Dependability attributes for increased security in component-based software development. Journal of Computer Science, 10(8), 1298–1306. doi:10.3844/jcssp.2014.1298.1306.

[97] Kahtan, H., Bakar, N. A., & Nordin, R. (2014). Awareness of embedding security features into component-based software development model: A survey. Journal of Computer Science, 10(8), 1411–1417. doi:10.3844/jcssp.2014.1411.1417.

[98] Kahtan, H., Bakar, N. A., & Nordin, R. (2014). Embedding Dependability Attributes into Component-Based Software Development Using the Best Practice Method: A Guideline. Journal of Applied Security Research, 9(3), 348–371. doi:10.1080/19361610.2014.913230.

[99] Keele, S. M., & Bell, R. C. (2008). The factorial validity of emotional intelligence: An unresolved issue. Personality and Individual Differences, 44(2), 487-500. doi:10.1016/j.paid.2007.09.013.

[100] Al-Ahmad, A. S., & Kahtan, H. (2018). Cloud Computing Review: Features and Issues. 2018 International Conference on Smart Computing and Electronic Enterprise, ICSCEE 2018. doi:10.1109/ICSCEE.2018.8538387.

[101] Rajak, A. A. (2022). Emerging technological methods for effective farming by cloud computing and IoT. Emerg. Sci. J., 6(5), 1017-1031. doi:10.28991/ESJ-2022-06-05-07.

[102] McLean, R. S., Antony, J., & Dahlgaard, J. J. (2017). Failure of Continuous Improvement initiatives in manufacturing environments: a systematic review of the evidence. Total Quality Management and Business Excellence, 28(3–4), 219–237. doi:10.1080/14783363.2015.1063414.

[103] Golder, S., Loke, Y. K., & Zorzela, L. (2014). Comparison of search strategies in systematic reviews of adverse effects to other systematic reviews. Health Information & Libraries Journal, 31(2), 92–105. doi:10.1111/hir.12041.

[104] Pucher, K. K., Boot, N. M. W. M., & De Vries, N. K. (2013). Systematic review: School health promotion interventions targeting physical activity and nutrition can improve academic performance in primary - and middle school children. Health Education, 113(5), 372-391. doi:10.1108/HE-02-2012-0013.

[105] Hu, Y., & Bai, G. A systematic literature review of cloud computing in eHealth. arXiv preprint, arXiv:1412.2494. doi:10.48550/arXiv.1412.2494.

[106] Mainka, C., Somorovsky, J., & Schwenk, J. (2012, June). Penetration testing tool for web services security. 2012 IEEE Eighth World Congress on Services. doi:10.1109/SERVICES.2012.7.

[107] Xing, B., Gao, L., Zhang, J., & Sun, D. (2010, October). Design and implementation of an XML-based penetration testing system. 2010 International Symposium on Intelligence Information Processing and Trusted Computing. doi:10.1109/IPTC.2010.109.

[108] Deptula, K. (2013). Automation of cyber penetration testing using the detect, identify, predict, react intelligence automation model. Master Thesis, Naval Postgraduate School Monterey, Monterey, United States.

[109] Halfond, W. G., Choudhary, S. R., & Orso, A. (2009, April). Penetration testing with improved input vector identification. 2009 International Conference on Software Testing Verification and Validation. doi:10.1109/ICST.2009.26.

[110] Jones, G. (2013). Penetrating the cloud. Network Security, 2013(2), 5–7. doi:10.1016/S1353-4858(13)70028-X.

[111] Byeong-Ho, K. A. N. G. (2008). About Effective Penetration Testing Methodology. Journal of Security Engineering, 5(5), 425-432. (In Korean).

[112] LaBarge, R., & McGuire, T. (2013). Cloud penetration testing. arXiv preprint, arXiv:1301.1912. doi:10.5121/ijccsa.2012.2604.

[113] Halfond, W. G. J., Choudhary, S. R., & Orso, A. (2011). Improving penetration testing through static and dynamic analysis. Software Testing Verification and Reliability, 21(3), 195–214. doi:10.1002/stvr.450.

[114] Baškarada, S. (2014). Qualitative case studies guidelines. The Qualitative Report, 19(40), 1-25.

[115] Hemmati, H., Briand, L., Arcuri, A., & Ali, S. (2010). An enhanced test case selection approach for model-based testing: an industrial case study. Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering. doi:10.1145/1882291.1882331.

[116] Yin, R. K. (2011). Applications of case study research. SAGE, London, United Kingdom.

[117] Shelke, A., & Mehendale, N. (2023). A CNN-based android application for plant leaf classification at remote locations. Neural Computing and Applications, 35(3), 2601-2607. doi:10.1007/s00521-022-07740-1.

[118] Li Li, R., Abendroth, D., Lin, X., Guo, Y., Baek, H. W., Eide, E., ... & Van der Merwe, J. (2015). Potassium: penetration testing as a service. Proceedings of the sixth ACM symposium on cloud computing. doi:10.1145/2806777.2806935.

[119] Cheah, M., Shaikh, S. A., Bryans, J., & Wooderson, P. (2018). Building an automotive security assurance case using systematic security evaluations. Computers and Security, 77, 360–379. doi:10.1016/j.cose.2018.04.008.

[120] Kamara, S., Fahmy, S., Schultz, E., Kerschbaum, F., & Frantzen, M. (2003). Analysis of vulnerabilities in internet firewalls. Computers & Security, 22(3), 214–232. doi:10.1016/S0167-4048(03)00310-9.

[121] Goseva-Popstojanova, K., & Perhinschi, A. (2015). On the capability of static code analysis to detect security vulnerabilities. Information and Software Technology, 68, 18–33. doi:10.1016/j.infsof.2015.08.002.

[122] Al-Azzani, S., Al-Natour, A., & Bahsoon, R. (2014). Architecture-centric testing for security: An agile perspective. Agile Software Architecture. Morgan Kaufmann, Burlington, United States. doi:10.1016/C2012-0-01208-2.

[123] Mouelhi, T., Le Traon, Y., Abgrall, E., Baudry, B., & Gombault, S. (2011). Tailored shielding and bypass testing of web applications. 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. doi:10.1109/ICST.2011.56.

[124] Ahmad, A. A.-S., Brereton, P., & Andras, P. (2017). A Systematic Mapping Study of Empirical Studies on Software Cloud Testing Methods. 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). doi:10.1109/qrs-c.2017.94.

[125] Pandey, A., & Mishra, S. (2021). Does the executive perception of the Value of information technology (IT) influence the IT strategy? A case Study. Journal of Information Systems Applied Research, 14(1), 24-35.

[126] Ceric, A., & Holland, P. (2019). The role of cognitive biases in anticipating and responding to cyberattacks. Information Technology & People, 32(1), 171-188. doi:10.1108/ITP-11-2017-0390.

[127] Chung, S., Mueller, S., & Kim, J. Y. (2018). Architecture-Driven Penetration Testing against a Cyber-Physical System. Information Systems Education Journal, 16, 37–44.

[128] Shah, M. P. (2019). Comparative Analysis of the Automated Penetration Testing Tools. Master Thesis, National College of Ireland, Dublin, Ireland.

[129] Kahtan, H., Bakar, N. A., Nordin, R., & Abdulgabber, M. A. (2014). Evaluation dependability attributes of web application using vulnerability assessments tools. Information Technology Journal, 13(14), 2240-2249.

[130] Krein, J. L. (2014). Replication and Knowledge Production in Empirical Software Engineering Research. Ph.D. Thesis, Brigham Young University, Provo, United States.

[131] Livshits, V. B., & Lam, M. S. (2005, August). Finding Security Vulnerabilities in Java Applications with Static Analysis. 14th USENIX Security Symposium, 1-5 August, 2005, Baltimore, United States.

[132] Yohanandhan, R. V., Elavarasan, R. M., Pugazhendhi, R., Premkumar, M., Mihet-Popa, L., Zhao, J., & Terzija, V. (2022). A specialized review on outlook of future Cyber-Physical Power System (CPPS) testbeds for securing electric power grid. International Journal of Electrical Power & Energy Systems, 136, 107720. doi:10.1016/j.ijepes.2021.107720

[133] Hurst, W, Shone, N, El Rhalibi, A, Happe, A, Kotze, B & Duncan, B (2017) Advancing the Micro-CI Testbed for IoT Cyber-Security Research and Education. The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 19-23 February, 2017, Athens, Greece.

[134] Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010). State of the Art: Automated Black-Box Web Application Vulnerability Testing. 2010 IEEE Symposium on Security and Privacy. doi:10.1109/sp.2010.27.

[135] Khoury, N., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011). An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection. 2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing. doi:10.1109/passat/socialcom.2011.199.

[136] Mannino, J. (2012). OWASP Goatdroid project. GitHub, Inc. Available online: https://github.com/nvisium-jack-mannino/OWASP-GoatDroid-Project (accessed on April 2023).

[137] Bures, M., Herout, P., & Ahmed, B. S. (2020). Open-source Defect Injection Benchmark Testbed for the Evaluation of Testing. 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal. doi:10.1109/icst46399.2020.00059.

[138] Lindvall, M., Rus, I., Shull, F., Zelkowitz, M., Donzelli, P., Memon, A., Basili, V., Costa, P., Tvedt, R., Hochstein, L., Asgari, S., Ackermann, C., & Pech, D. (2005). An evolutionary testbed for software technology evaluation. Innovations in Systems and Software Engineering, 1(1), 3–11. doi:10.1007/s11334-005-0007-z.

[139] Ridene, Y., & Barbier, F. (2011). A model-driven approach for automating mobile applications testing. Proceedings of the 5th European Conference on Software Architecture: Companion Volume. doi:10.1145/2031759.2031770.

[140] Cadar, C., Godefroid, P., Khurshid, S., Păsăreanu, C. S., Sen, K., Tillmann, N., & Visser, W. (2011). Symbolic execution for software testing in practice. Proceedings of the 33rd International Conference on Software Engineering, 1066–1071. doi:10.1145/1985793.1985995.

[141] Yuhong Cai, Grundy, J., & Hosking, J. (2004). Experiences integrating and scaling a performance test bed generator with an open source CASE tool. Proceedings. 19th International Conference on Automated Software Engineering, 2004. doi:10.1109/ase.2004.1342722.

[142] Hooda, I., & Singh Chhillar, R. (2015). Software Test Process, Testing Types and Techniques. International Journal of Computer Applications, 111(13), 10–14. doi:10.5120/19597-1433.

[143] D'Angelo, G., Ficco, M., & Palmieri, F. (2020). Malware detection in mobile environments based on Auto encoders and API-images. Journal of Parallel and Distributed Computing, 137, 26–33. doi:10.1016/j.jpdc.2019.11.001.

[144] Al Nasseri, H. M. (2019). Detecting cloud virtual network isolation security for data leakage. Ph.D. Thesis, University of St Andrews, St Andrews, United Kingdom.

[145] Valdi, A., Lever, E., Benefico, S., Quarta, D., Zanero, S., & Maggi, F. (2015). Scalable Testing of Mobile Antivirus Applications. Computer, 48(11), 60–68. doi:10.1109/MC.2015.320.

[146] Schwarz, K., Schwarz, F., & Creutzburg, R. (2020). Conception and implementation of professional laboratory exercises in the field of open source intelligence (OSINT). Electronic Imaging, 32(3), 278-1-278–10. doi10.2352/issn.2470-1173.2020.3.mobmu-278.

[147] Holibaugh, R., Perry, J. M., & Sun, L. A. (1988). Phase I Testbed Description: Requirements and Selection Guifdelines. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, United States.

[148] Wermelinger, M., Yu, Y., Lozano, A., & Capiluppi, A. (2011). Assessing architectural evolution: A case study. Empirical Software Engineering, 16(5), 623–666. doi:10.1007/s10664-011-9164-x.