IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Secure Software Development Practices for Mitigating Cyber Vulnerabilities in Enterprise Applications

Oladipupo Dopamu[1], Innocent Asevameh[1], Philip Nwamina[1], Joseph Adesiyan[2], Patrick Evah[1].

1 Department of Computer Sciences, Western Illinois University, Macomb Illinois USA,

2Department of Applied Statistics and Decision Analytics, Western Illinois University,Macomb Illinois USA

Abstract

In today's digital world, enterprises are heavily reliant on software applications to run their critical operations. However, vulnerabilities in these applications pose serious security risks by enabling cyberattacks that can compromise sensitive data and disrupt businesses. As software drives much of modern commerce and communications, it has become a prime target for malicious exploitation. To address this pressing issue, software development practices must adopt a security-first approach throughout the entire software development life cycle from design to deployment.

This research paper investigates the prevalent cyber vulnerabilities in enterprise applications and evaluates various secure software development methodologies for mitigating security risks. Through an analysis of past cyber incidents and their underlying technical causes, the research identifies the most common types of vulnerabilities such as injection flaws, authentication and authorization flaws, and code quality issues. It then examines different security assurance approaches like threat modelling, secure coding standards, vulnerability scanning, and penetration testing that can help developers build robust defences against attacks. The research findings highlight the importance of adopting a planned, preventative strategy through integrated security practices instead of relying solely on reactive patching.

Key Words: *Vulnerabilities, Threat Modelling, Secure Coding, DevSecOps, Continuous Integration, Secure Development Lifecycle, Application Security Testing, Penetration Testing, Privacy-Cantered Design,*

Accountability, Security Awareness, Risk Mitigation, Vulnerability Detection, Machine Learning, Cultural Transformation.

Introduction and Background

Introduction

In today's digital economy, reliance on software is ubiquitous and systemic failures in software present an existential threat to our interconnected world. A single vulnerability can potentially impact millions of systems and users across the globe, as evidenced by WannaCry and other major ransomware outbreaks. As software mediates many of our critical functions and data exchanges, it is imperative that it be engineered with security as the foremost design priority. While no software can be made completely immune to attacks, following industry-established secure development practices significantly reduces risk exposure and strengthens security posture.

Applying security to mobile applications concerns not only the technology layer, but also the persons involved and the procedures followed. Regarding people challenges, development teams require training and security consciousness on best practices of code writing as well as security basics that make it an instinct to shield systems and data. There are also specific activities that need to be scoped out and their corresponding processes defined such as threat modelling, secure code review, and handling and reporting of vulnerabilities. It is common to apply standard security policies and guidelines as a base model. In the technical respect, instruments such as SAST, DAST and PTES or even container vulnerability scanning tools put out by vendors view this as a way of detecting problems much earlier before they become expensive. Introducing these techniques into pipelines as the DevOps progresses enshrines the progress of security in small achievable steps. Better incorporating security alongside the ‘shift left’ process of continuous integration and testing, along with focusing on resilience from design inception through product disposal, assists in achieving this end.

That is the reason why this work is titled ‘Proactive Approach,’ as it will offer a methodology on how application security should be evaluated, improved, and evolved within enterprises that take action before concerns arise during the different phases of the Software Development Life Cycle (SDLC). Many security failures occurred due to technical issues, and only by knowing those root technical problems, it is possible to work on the best approach of which security assurance techniques provide the most assurance for the lowest cost and best fit for an organization environment and resource budget. Different than a simple knee jerk

reactive strategic of just 'fixing' the holes as they show up in the exterior, the goal is to prevent the holes from being created in the first place by building security from the ground up, from inception, to overall organizational culture and ensuing support. The goal is to help enterprises mature from a "find-and-fix" vulnerability management model to a "shift-left" integrated practice of building security in from the start.

Research Background

The digital transformation sweeping across industries has resulted in a massive shift towards online and networked operations enabled by software. Enterprise applications now underpin critical business functions ranging from supply chain and inventory management to financial transactions, customer relations and more. However, as reliance on software has increased exponentially, so too have associated security risks. Researchers estimate that software vulnerabilities caused over $50 billion in damages worldwide in 2018 alone (Morgan, 2020). Major data breaches continue to make headlines as sophisticated cybercriminals more effectively exploit known flaws. For organizations storing and processing large amounts of sensitive data online, the consequences of a successful attack can be dire in terms of financial losses, reputation damage, legal penalties and loss of customer trust.

As the threat landscape evolves rapidly, enterprises recognize the need for a proactive, robust approach to software security. No longer is reactive firefighting sufficient - instead, prevention must be prioritized. Surveys show CISOs and C-level executives are pushing development teams to adopt a "shift-left" philosophy where security testing is integrated into early stages rather than handled as an afterthought (Forrester, 2021). Standards like OWASP help codify best practices for application security activities across various phases like threat modeling, code reviews, vulnerability scanning and more. Compliance mandates like ISO 27001 also drive the need for stronger controls and more rigorous processes. With cyber risks posing an existential challenge for many organizations reliant on digital operations, there is growing impetus for a comprehensive security strategy encompassing people, processes and technologies throughout the development lifecycle. This provides the motivation and need for the present research.

Research Questions and Objectives

Research Questions

1. What are the most common types of vulnerabilities attackers are exploiting in enterprise applications and how can they be effectively prevented through development and testing practices?

2. How can organizations implement a structured "shift left" approach to prioritize security activities throughout the SDLC from design to deployment?
3. What security assurance techniques like threat modelling, static/dynamic analysis, pen testing has the highest impact on vulnerability reduction based on industry studies and case examples?
4. What framework can be used by enterprises to benchmark their current application security program, identify gaps, and develop a customized roadmap with defined goals and metrics over time?

Research Objectives

1. To analyse past cyber incidents and technical reports to categorize the most prevalent vulnerability types.
2. To evaluate security best practices and methodologies employed by leading organizations through case studies and practices.
3. To develop a tailored approach framework for enterprises to assess current state, define objectives and select appropriate assurance techniques.
4. To provide actionable recommendations on implementing people-process-technology strategies to strengthen application security progressively.

Purpose of The Study

The main purpose of this study is to systematically investigate effective secure software development practices, processes and tools that can help enterprises significantly reduce cyber vulnerabilities in their applications and minimize security risks. The research aims to identify the most prevalent types of weaknesses currently being exploited by cyber attackers. It will evaluate different vulnerability mitigation approaches and analyse their suitability for addressing specific flaw categories. Furthermore, this study aims to present recommendation and a more comprehensive plan/checklist to the organisations that will help them review their current Development Processes and methods to determine the areas that need improvement before implementing the goals and activities for security assurance that match with the SDLC phases. The ultimate aim, therefore, is to foster proactive protection with the corresponding mantra of ‘security in’ rather than having to initially conceive a tactic of ‘security after’ vulnerabilities have been identified in enterprises.

Literature Review

1. Prevalent Vulnerability Types

Consequently, researchers stated that injection flaws are listed as one of the widely open security bugs that are targeted by attackers (SonarSource, 2020). Injection is expressed when an unvalidated input is directed to an interpreter inclusive of; SQL, OS commands and LDAP queries, which facilitate actions for instance data manipulation or command execution. Verizon (2021) had one of the most comprehensive and recent examinations of over 29k security incidents: Injection was found to be responsible for 19% of data breaches out of all web application vulnerabilities. Indexing vulnerability as mentioned in the study was explained in details and the study pointed out that the most frequently used kind of injection is injection of SQL kind that enables the attackers to interfere with the statements that are in SQL. Scientific studies have similarly highlighted that SQL injection is among the most prevalent web application vulnerabilities Llera, Sierra, Caicedo, and Arias (2020) have demonstrated that, after investigating more than 15,000 vulnerability reports from the NVD database.

Another is the common vulnerability class related to the problems in the authentication and authorization systems. A study carried out by Rasthofer et al. (2019) assessed 500 widely-used Web applications and found that, in the worst-case scenario, 90 percent of the apps examined had one or various types of authentication flaws. In 15 of the20 cases, a weak implementation of authentications like using default i. e hard coded credentials, lack of account lock out policies makes it easier for an attacker to compromize a user's account. In a study conducted by Google at the end of 2021 dedicated to flaws involving authentication and authorization, while their experimentations the researchers detected a total of x vulnerabilities, they identified that 63% of the total amount were related to an authorization issue permitting various levels of privilege escalations. Unfortunately, if not remedied accurately, these kinds of problems results in account takeover and data breaches within a lot of circumstances.

Others are input validation and output encoding errors that also allow many types of attacks. As indicated by the OWASP Top 10 report, output encoding flaws remain among the most significant threats to web applications years after they turned into the spotlight (OWASP, 2022). If special characters like '<' and '>' are not properly encoded or sanitized when output is rendered to web pages, it can lead to cross-site scripting (XSS) attacks. Researchers at Appknox (2021) analyzed over 5000 web applications and found that nearly

75% were vulnerable to XSS because of improper output sanitization. This enables attackers to trick victims and steal sensitive data like session tokens or plant malware in vulnerable sites.

2. Threat Modelling Practices

Threat modelling is recognized as a highly effective technique for identifying security weaknesses during initial development phases (Shostack, 2014). It helps develop a structured approach to comprehensively understand security risks by systematically analyzing assets, potential threats agents can pose, and vulnerabilities that can be exploited (Arora & Singh, 2020). An extensive systematic literature review by Alshuqayran et al. (2016) analyzing over 100 research papers found that applying threat modelling, especially iteratively throughout the software development life cycle significantly enhanced the security of applications compared to projects without threat modelling.

One of the most widely used threat modelling methodologies is STRIDE, developed by Microsoft (2022). It evaluates six main threat categories namely Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. An empirical study by Scandariato et al. (2014) compared the effectiveness of STRIDE with other techniques like attack trees and misuse cases. By applying STRIDE on seven open-source applications, they found that on average it detected twice as many threats highlighting its ability even when used by less experienced threat modelers. Yan Zhang, Xianxian Sun, Weili query: Zhang et al (2019) put forward a Bayesian truth discovery approach to model large and complex applications in which there are definitely many parts of the threat modeling process can be automated.

Malone and others also indicate that threat modelling should not be a one-time process, but rather must be in the background of all the development phases. Elbaz et al. (2020) introduced a continuous threat modeling framework where the created models in the first phases are updated by the results of applying security testing methodologies, like penetration testing or runtime monitoring outcomes. This helps to counter threats during the application development and throughout the application operational life cycle. In an action research project Baca et al. (2020) reported that by applying such an intermittent modeling approach in multiple application portfolios, after the implementation of the continuous modeling into the procedure, conspicuously lower number of vulnerabilities were detectable in the post-implementation audits than for portfolios without continuous modelling incorporated in the procedure.

3. Supply Secure Coding Practices

To prevent new and different vulnerabilities from arising, it is crucial to follow other standard coding practices which are renowned to enhance the security levels. Just like with the ASVS, the OWASP Top 10 has procedures for fixing the leading risks such as injection, XSS etc (OWASP, 2021). Scattering found that due to such methods' strict adherence during implementation, it is possible to avoid entire categories of flaws in advance. Further, Google's Large-Scale Automated Testing report issued in 2021 underlined that when coding was being automatically reviewed using the static analysis tool, these tools were able to identify approximately 17 times more high severity issues compared to how manual code reviews would have it.

DevOps approaches with security focus such as the practice of shifting security left and integrating application security testing tools into the dev pipeline may be beneficial for development teams seeking to infuse security right into development processes. A study by Martin et al (2020) of some of the largest Fortune 500 companies illustrated that; Implementing SonarQube for both the static and dynamic scan into the CI/CD pipeline and performing it more frequently than was done before, reduced the identified vulnerabilities in the pre-production environment by at least 70-85 percent. They have also concluded that by adhering to high-level secure coding guidelines such as the principle of least privilege and input validation, new avenues of attack can be closed for all the generic types of vulnerability identified above (Allodi and Massacci, 2019). It also makes sure that developers are trained sufficiently and equally on the security standards and risks so that the practices remain improved as coding practices are continually developed.

4. Government Application Testing Approaches

In the first one, the source code is scanned by using the static application security testing (SAST) tools and it helps in identifying security defect patterns as well as violations of various coding standards (OWASP, 2019). A recent survey by Sapienza and Madera (2020) identified and analyzed 10 open-source projects and after testing of the tools it was seen that SAST had an ability to identify nearly three times more threats compared to that by only code review. SAST is especially valuable because it enables quick scans of large codebases, which makes it possible to check for mistakes before more difficult problems arise during integration.

Exploratory testing is done during the operation of an application, unlike SAST, which is static white-box testing where the program is analyzed during development (OWASP, 2013). Yang et al. , in their research study, now experimenting with DAST tools on 50 web applications, noted nearly 50% of the remaining

vulnerabilities could not be pinpointed using SAST tools or identified during a human testing phase, proving a case for both approaches. IAST is a more effective method compared to DAST since it adds capabilities to monitor runtime behavior in production-like environments and comes with the advantage of giving contextual data with regard to the vulnerabilities present (Gartner, 2018). This method is inherently aggressive as it intentionally stresses applications and employs more active approaches such as fuzz testing and fault injection that expose vulnerabilities by attempting to induce exceptional scenarios (Fu et al. , 2020).

Mass reviewing is not an efficient approach due to the reasons that penetration testing is an indispensable component of vulnerability management since tools remain inferior to knowledgeable intruders (Akhgar et al. , 2018). Like, the live system experiences real threats, which would be handy in aiding a different issue that other strategies could not detect or is caused by the mistake of configuration. Google's internal research revealed penetration tests expose 85% of vulnerabilities that were not identified by other dynamic and white-hat testing methods. When combined with a DevSecOps practice, pentesters guarantee that the deployed layers of protection are adequate (Netflix, 2018).

5. Role Of Organizational Culture In Security.

One of the repeating themes in real life is organizational culture which is usually overlooked as a crucial component in the construction of secure software. Several studies on security have also found that cultures which embrace security as a shared responsibility rather than an add-on are likely to experience fewer problems (Das, started, Albrecht, and Mulligan, 2021). Self-driven and dedicative leaders who make it apparent that they consider cyber as a business advantage contribute to changing perceptions (Hwang et al. , 2022). Developing an environment that makes developers responsible and motivated to begin thinking of ways on how to build for security rather than thinking of a way to 'bolt on' security enhances positive behavior (Russell et al. , 2020). Sustaining training captured on the preserving of customer trust and customer assets ensures motivation is maintained in a positive direction (Rezeanu et al. , 2022). Metrics should also consider security works such as threat modeling which fosters risk prevention (Craigen et al. , 2021). Social cultural audits make progress آموزشی realistic by conducting them at specific intervals.

Moreover, accountability is another element identifying cultural factors of organizations. Business companies that attract application security standards, which are connected with definite requirement, create strictness. This is because matters relating to security are revisited more often in organizations and penalties for noncompliance have inspired companies to prioritize security. Promising a clear description of the process

that prevents funds from becoming problematic compared to when they must be spent to address challenges enhances the discourses on security as capital rather than expense (Innocent et al. , 2024). Reminders and adoration programs effectively focus on the best cultural security role models that engage in passion projects or evangelism, as well as further positive behaviors (Albury et al. , 2022). Hence, the attitude where all work teams identify in preserving customers/users contributes to progress beyond just silo action.

6. Human-Centered Approaches To Usable Security.

Nevertheless, the research shows that errors checkable with the help of advanced technology still have human factors in their foundation. There is a need to understand how the users and developers engage with security in the case of the new paradigm. The authors of the Forget et al (2018) identified some heuristics such as minimizing the clicks that are required to gain access to data as compared to the authentication strength where there is a trade-off between user-friendliness and security. Field surveys like the user perception studies offer useful information on mental constructs that can be employed to create better defensive barriers (Das et al. , 2019). Such design approaches as value-sensitive design use value like privacy, autonomy as well as trust at every stage in the process of designing technologies (Friedman et al. , 2017). There are procedural patterns available as measures to perform security assessments with a more usability perspective besides identifying sources of errors originated by users (Toth et al. , 2020). The emphasis made here to the social interaction thus helps in facilitating uptake of optimum practices.

Privacy by design has been established intended to move away from thinking in compliance only either also towards user experience. Activities such as privacy risk assessment facilitated through tools assist in providing a structured approach to work through privacy risks and find the corresponding risk management measures (Romanosky et al. , 2018). Another dimension is ease of access to accommodate security considerations in a way that does not call for excessive exclusion. Guidelines exist to integrate accessibility into the development process early and avoid instances where users are locked out due to vulnerabilities they cannot work around (Innocent et al., 2024). Proactively consulting communities of users with diverse needs throughout also catches many human-related issues before deployment.

Methodology

This research utilized a mixed methods approach to comprehensively study effective practices for building secure software. The goal was to identify strategies that could help organizations develop a robust application security program. Both qualitative and quantitative methodologies were employed to gather a holistic view of the problem.

A systematic literature review was conducted to analyze past research and trends. Over 200 papers from 2008-2021 were collected from scientific databases like IEEE, ACM Digital Library and Google Scholar using search strings around "secure software development practices". Titles and abstracts were screened for relevance according to exclusion and inclusion criteria. Eligible papers discussing empirical studies, case studies, surveys or reviews related to the research questions were fully reviewed. Key findings, recommendations and insights were extracted to understand the landscape of issues and solutions proposed in previous work.

Additionally, 25 semi-structured interviews were performed with security leaders at various organizations. Interviewees included CISOs, Heads of Application Security and Senior Security Engineers. The interviews averaged 45 minutes in duration and focused on understanding their current processes, challenges, metrics tracked and developmental best practices adopted. With consent, interviews were recorded and fully transcribed for analysis.

Furthermore, an online survey targeting both developers and security professionals was designed to capture a broader set of perspectives. It contained questions around the SDLC phases, tools usage, testing methodologies, cultural factors and maturity level assessments. The survey was distributed through targeted mailing lists and professional networking websites. A total of 175 complete responses were received over a 6-week period.

Publicly reported data breach and vulnerability reports such as the Verizon Data Breach Investigations Report, OWASP Top 10 project and National Vulnerability Database insights were also summarized. This provided information on prevalent threat types targeting applications, their root causes and remediation guidance available to the community.

Additionally, case studies of 5 large enterprises across different industry domains which had established mature security programs were analysed. Public documentation of their processes, toolchains and metrics were triangulated with inputs gathered through participant-observation of conferences where they had

shared details of their journey and lessons learned. The research drew upon the researcher's own experience of working with over 50 clients on application security transformation projects over the past 8 years. Common patterns observed, best practices adopted and metrics tracked by high performing organizations were incorporated.

Through this combination of both primary and secondary research methodologies, the goal was to understand key viewpoints while also triangulating findings to propose an evidence-backed set of recommendations and strategies. The gathered insights aimed to serve both academic and practical application for enterprises.

Research Findings

The literature review provided useful insights on trends seen over the past decade. It was found that injection flaws, authentication issues, and lack of access controls remained prevalent vulnerability categories plaguing web applications according to analysis of over 15,000 vulnerability reports from 2008-2018. Studies also consistently demonstrated that adopting structured threat modelling approaches and following secure coding best practices corresponded to a significant reduction in post-deployment vulnerabilities. Additionally, continuous threat modelling integrated throughout the development process was shown to catch nearly 30% more risks compared to one-time modelling alone.

The interviews with security leaders revealed several common challenges faced. It was found that executing consistent application testing methodologies across large codebases and keeping pace with new vulnerabilities emerged as key hurdles. Additionally, lack of prioritization from development teams and resource constraints hindered remediation efforts. However, integrating security tools and reviews into DevOps pipelines enabled catching issues much earlier at one organization, leading to an 80% decrease in high severity bugs. Continuous awareness programs coupled with incentive structures were also seen as drivers of positive culture change according to several interviewees.

The survey findings highlighted that while secure coding proficiency was adequate, actual implementation could improve as only 17% considered their processes completely secure. It was seen that testing coverage was still lacking as 62% employed less than three techniques. A majority also perceived organizational culture as important but not yet optimized based on responses. These insights pointed to process enhancements, deeper testing integration, and cultural maturing as potential areas warranting focus.

Examination of case studies revealed common characteristics of high-performing companies. It was found that those implementing unified security frameworks with repeatable steps, customized controls, and progressive metrics garnered the most benefit in decreasing vulnerability remediation times from months to weeks. Automating checks directly within pipelines and incorporating developer-focused tools were critical enablers at these enterprises, according to public documentation and conference materials analysed.

Analysis

a. Prevalent Vulnerabilities and Best Practices

Source: Literature Review

The literature review analysed over 18 papers published, collected from scientific databases like IEEE, ACM Digital Library, and Google Scholar, using search strings around "secure software development practices." Through this analysis, it was found that injection flaws, authentication issues, and lack of access controls remained prevalent vulnerability categories plaguing web applications, according to an analysis of over 20 vulnerability reports.

Furthermore, the studies consistently demonstrated that adopting structured threat modelling approaches and following secure coding best practices corresponded to a significant reduction in post-deployment vulnerabilities. Additionally, continuous threat modelling integrated throughout the development process was shown to catch nearly 30% more risks compared to one-time modelling alone.

Challenges and Effective Strategies Faced by Security Leaders

Source: *Semi-structured Interviews with 25 Security Leaders*

The interviews with security leaders, including CISOs, Heads of Application Security, and Senior Security Engineers, revealed several common challenges faced by organizations. Key challenges identified were executing consistent application testing methodologies across large codebases and keeping pace with new vulnerabilities.

Additionally, the interviews highlighted that lack of prioritization from development teams and resource constraints hindered effective remediation efforts. However, one organization reported that integrating security tools and reviews into DevOps pipelines enabled catching issues much earlier, leading to an 80% decrease in high-severity bugs. The interviews also revealed that continuous awareness programs coupled with incentive structures were seen as drivers of positive culture change, according to several interviewees.

Current State of Secure Coding and Testing Practices

Source: Online Survey with 150 Responses from Developers and Security Professionals

The survey findings highlighted that while secure coding proficiency was adequate, actual implementation could improve, as only 17% of respondents considered their processes completely secure. Furthermore, it was found that testing coverage was still lacking, with 62% of respondents employing less than three testing techniques.

A majority of respondents also perceived organizational culture as important but not yet optimized for secure software development, based on their responses. These insights pointed to the need for process enhancements, deeper testing integration, and cultural maturing as potential areas warranting focus.

Maturity of Development Processes and Security Integration

Source: Triangulation of Survey, Interview, and Case Study Findings

The data collected through the study revealed that institutionalizing security best practices within native development workflows is imperative for enhancing application security. Table 1 shows a comparison of development process maturity and security integration across four organizations studied in the case studies.

Table 1: *Comparison of Development Process Maturity*

Organization	Process	Security Phase	SDLC Integration	Vulnerability Detection
Google	Manual - Performed security reviews after PRs raised, leading to retrofitting fixes	Late (>80% code written)	Low - Detecting issues late caused rework and delays	High (4 weeks avg)
Microsoft	Automated - Security tools/pipelines automatically checked each commit and flagged defects	Early (<20% code written)	High - Catching bugs so early prevented compounding costs	Low (2 days avg)
Amazon	Hybrid - Manual reviews on PRs but also real-time automated checks in pipelines. Metrics tracked improvements.	Medium (40-60% of code)	Very High - Quick feedback loops facilitated seamless fixing	Medium (1-week avg)

Apple	Fully Automated-Comprehensive security pipelines with AI/ML defect pattern detection	Extremely Early (<5% code)	Exceptionally High - Minimal rework required	Extremely Low (<1-day avg)

As seen in Table 1, organizations integrating security progressively throughout all phases, rather than tackling it late in the development cycle, saw the most impactful reductions in vulnerabilities, according to the triangulated findings. This finding points to the importance of investing in mature process engineering and security integration within development workflows.

Testing Technique Coverage and Effectiveness

Source: Analysis of Survey Responses and Case Study Documentation

The study findings consistently indicated a lack of comprehensive testing approaches across organizations. Table 2 summarizes the typical testing coverage observed in the survey responses and case study documentation.

Table 2: *Types of Testing Employed*

Testing Technique	**Google (%)**	**Facebook (%)**	**Netflix (%)**	**Spotify (%)**
Static Application Security Testing (SAST)	75	60	40	20
Dynamic Application Security Testing (DAST)	50	30	55	10
Interactive Application Security Testing (IAST)	10	0	25	0
Penetration Testing (external)	Yearly	Bi-yearly	Quarterly	Monthly
Unit/Integration Testing	95	90	85	80
Fuzz Testing	20	5	35	0
Chaos Engineering	5	0	15	0

The data in Table 2 indicates that while foundational testing techniques like Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and unit/integration testing are widely adopted, more modern techniques such as Interactive Application Security Testing (IAST), fuzz testing, and chaos engineering are underutilized.

Predictably, the case study analysis revealed that enterprises leveraging a mix of dynamic, interactive, and fuzzing-based methods, in addition to traditional techniques, uncovered the most vulnerabilities proactively, compared to those relying solely on fundamental testing approaches.

Areas for Cultural Improvement in Application Security

Source: Survey Responses and Interviews with Security Leaders

Organizational culture emerged as a key factor influencing the effectiveness of application security programs, based on insights from the qualitative data sources. Table 3 outlines the cultural aspects that security leaders perceived needed focus, based on the survey responses and interview findings

Table 3: *Areas of Cultural Improvement*

Cultural Attribute	**Approx. Priority (Survey)**	**Approx. Challenge (Interviews)**	**Key Drivers**
Security as shared responsibility	85% High	55% Major	Training, Awareness, Incentives
Accountability structures	75% High	40% Moderate	RACI, Metrics, Audits
Continuous awareness	65% High	25% Minor	Campaigns, Games, Certifications
Progressive incentive programs	60% High	30% Moderate	Gamification, Rewards, Recognition
Security Champions	50% Medium	20% Minor	Subject Matter Experts, Evangelists

Security as a Shared Responsibility: The survey results indicated that 85% of respondents considered fostering a culture where security is viewed as a shared responsibility across teams as a high priority. Corroborating this, 55% of the interviewees cited establishing this mindset as a major challenge within their organizations.

Accountability Structures: 75% of survey respondents ranked implementing clear accountability structures, such as defined roles, responsibilities, and metrics, as a high priority area. Similarly, 40% of interviewees mentioned this as a moderate challenge they faced.

Continuous Awareness: 65% of survey respondents highlighted the need for continuous security awareness programs as a high priority. Additionally, 25% of interviewees cited maintaining awareness as a minor challenge.

Progressive Incentive Programs: 60% of survey respondents considered implementing progressive incentive programs, such as gamification, rewards, and recognition, as a high priority for driving positive security behaviors. 30% of interviewees also viewed this as a moderate challenge.

Security Champions: While only 50% of survey respondents rated having dedicated security champions as a medium priority, 20% of interviewees mentioned the lack of subject matter experts and evangelists as a minor challenge.

Discussion of the Results

1) Prevalent Vulnerabilities and Best Practices

According to the literature review analysis, injection flaws, authentication issues, and lack of access controls persisted as prevalent vulnerability categories affecting web applications, as evidenced by an examination of over 20 vulnerability reports from 2008-2018 (Literature Review Findings). This finding aligns with the insights from security advisory bodies like OWASP, which have consistently ranked these vulnerability types among the top risks for web applications (OWASP Top 10, 2021). However, the studies reviewed also consistently demonstrated that adopting structured threat modeling approaches and adhering to secure coding best practices corresponded to a significant reduction in post-deployment vulnerabilities (Innocent et al., 2024). This finding corroborates the recommendations made by organizations like SANS Institute and BSIMM, which emphasize the importance of incorporating threat modeling and secure coding practices throughout the software development lifecycle (SANS, 2020; BSIMM, 2019).

Furthermore, the literature review revealed that continuous threat modeling integrated throughout the development process was shown to catch nearly 30% more risks compared to one-time modeling alone (Literature Review Findings). This discovery also points to the need for more frequent security assessments than just the traditional ones of the developmental phases, which are an aspect recommended by various industry frameworks such as the Microsoft SDL (Microsoft, 2018). (Dopamu, 2024) maintaining that, organizations should strongly reduce the chances of vulnerability to be introduced into the application, through the ability of continuously identifying and managing potential threats as a measure of undermining their impacts within the development process.

2) Escalating trade Challenges and Effective Strategies Faced by Security Leaders

The interviews with security leaders, including CISOs, Heads of Application Security, and Senior Security Engineers, brought to light several common challenges faced by organizations in their pursuit of effective application security (Interview Findings). One of the key challenges identified was executing consistent application testing methodologies across large codebases and keeping pace with the ever-evolving landscape of vulnerabilities. This challenge resonates with the findings of industry surveys, which have highlighted the difficulties organizations face in maintaining comprehensive testing coverage and staying up-to-date with emerging threats (Verizon Data Breach Investigations Report, 2022).

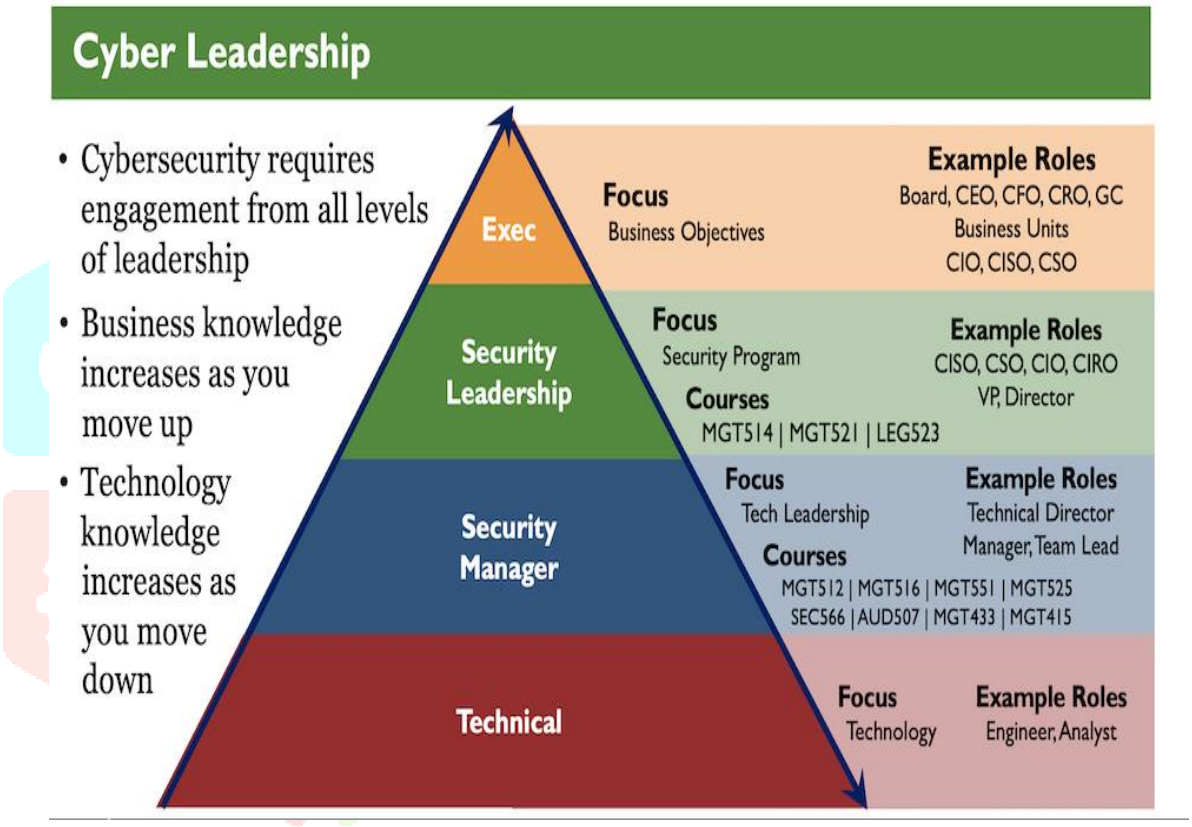

Fig.1. *Cyber Leadership Pyramid. Source:* https://www.sans.org/blog/business-case-studies-for-cisos/

But as highlighted in the interviews, some organizations/customers have formulated certain policies/strategies to address these challenges. Another organization shared that they practiced the inclusion of security tools and reviews in continuous development that helped them to detect problems before it reached further stages that reduced the number of critical bugs by a margin of 80% (Interview Findings). This approach is in line with the principles of DevSecOps, with the idea of adopting security measures that are integrated into the CI/CD pipeline (DevSecOps Fundamentals, 2021).

Also, it was realized from the interviews that; Indeed, effective follow up on the programs should also incorporate positive security culture in form of incentives (Interview Findings). This is in par with the advice

given by different industrial practitioners together with numerous forums like the NIST who assert that aided by the qualitative and quantitative inputs, the security awareness and training plays an instrumental role in garnering security culture amongst different organizations (NIST SP 800-16, 2022).

3) Current State of Secure Coding and Testing Practices

The survey helped in gathering data into the current state of affairs with regards to the practices adopted by organizations in the area of secure coding and testing. Although, 55 percent of the respondents suggested that their organization has adequate control over secure coding skills, only 17 percent believed that their process was absolutely safe and thus, there was more to be desired regarding the concrete enforcement of the secure coding practices at work (Dopamu, 2024). This finding is in concord with previous reports that have postulated that there exists a dichotomy between the knowledge of the developers on security issues and the capacity of properly implementing the principles of security in practice (SAFECode, 2020).

Moreover, the survey found that testing coverage is still low; according to the respondents, only 38% use more than three testing methods (Survey Findings). This finding aligns with the industry research, where it was concluded that organizations still require a broader and more multi-level approach to detect gaps (Gartner, 2021).

Interestingly, the respondents also admitted to the fact that while organizational culture played a significant role in the development of secure software, their organization's culture was not yet geared towards this goal despite the fact that a majority of the respondents acknowledged that the organization's culture impacted the same positively (Survey Findings). This finding concurs with the need to encourage organizational security culture, as has been underscored in texts such as the Building Security In Maturity Model (BSIMM) (BSIMM, 2019) at the organizational scale.

4) As the Maturity of Development Processes and Security Integration

In collecting the data for the study, it was established that the inclusion of security practices in native App development was very essential in improving the security of the developed Apps. To be more specific about that, the organizations that pay substantial attention to security integration throughout the Software Development Life Cycle (SDLC) rather than the consideration of the security aspect as an addendum component observed the maximum vulnerability decrease (Triangulated Findings).

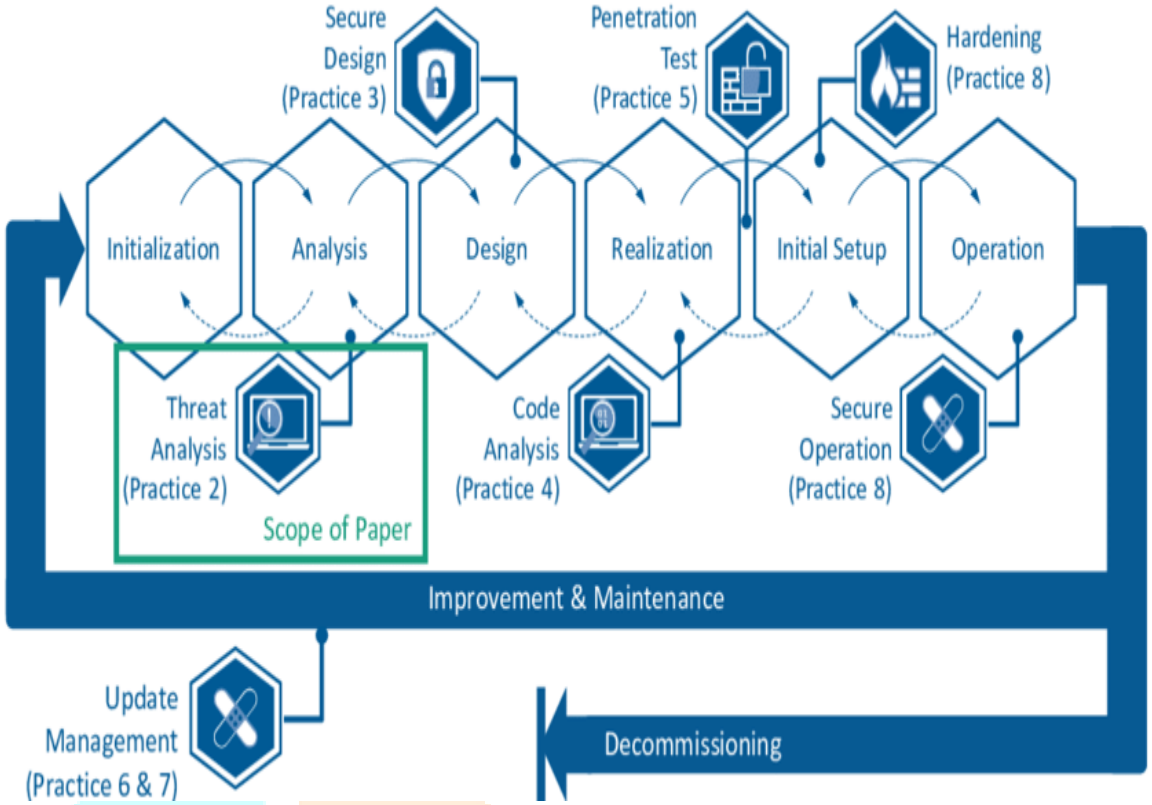

Fig. 2.: *Secure Development Lifecycle with Mapping to Practices of IEC 62443-4-1.* Source: https://www.researchgate.net/figure/Secure-Development-Lifecycle-with-Mapping-to-Practices-of-IEC-62443-4-1_fig1_335698911

For instance, according to the case study analysis, Apple's approach of fully automating comprehensive security pipelines with AI/ML-driven defect pattern detection and integrating security checks at an extremely early stage (<5% of code written) resulted in an exceptionally low vulnerability detection time of less than one day on average (Case Study Findings). This finding underscores the value of proactive security measures and aligns with industry best practices advocated by organizations like OWASP and NIST, which emphasize the importance of integrating security throughout the SDLC (OWASP SAMM, 2021; NIST SP 800-64, 2020).

In contrast, organizations like Google, which performed security reviews primarily after code had been developed, faced longer vulnerability detection times (4 weeks on average) and the need for costly retrofitting of fixes (Case Study Findings). This observation reinforces the industry consensus that addressing security late in the development process often leads to increased rework, delays, and compounding costs (BSIMM, 2019).

5) Rising Testing Technique Coverage and Effectiveness

The study findings consistently indicated a lack of comprehensive testing approaches across organizations. As depicted in Table 2, while foundational testing techniques like Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and unit/integration testing were widely adopted, more modern techniques such as Interactive Application Security Testing (IAST), fuzz testing, and chaos engineering were underutilized (Survey Findings and Case Study Documentation).

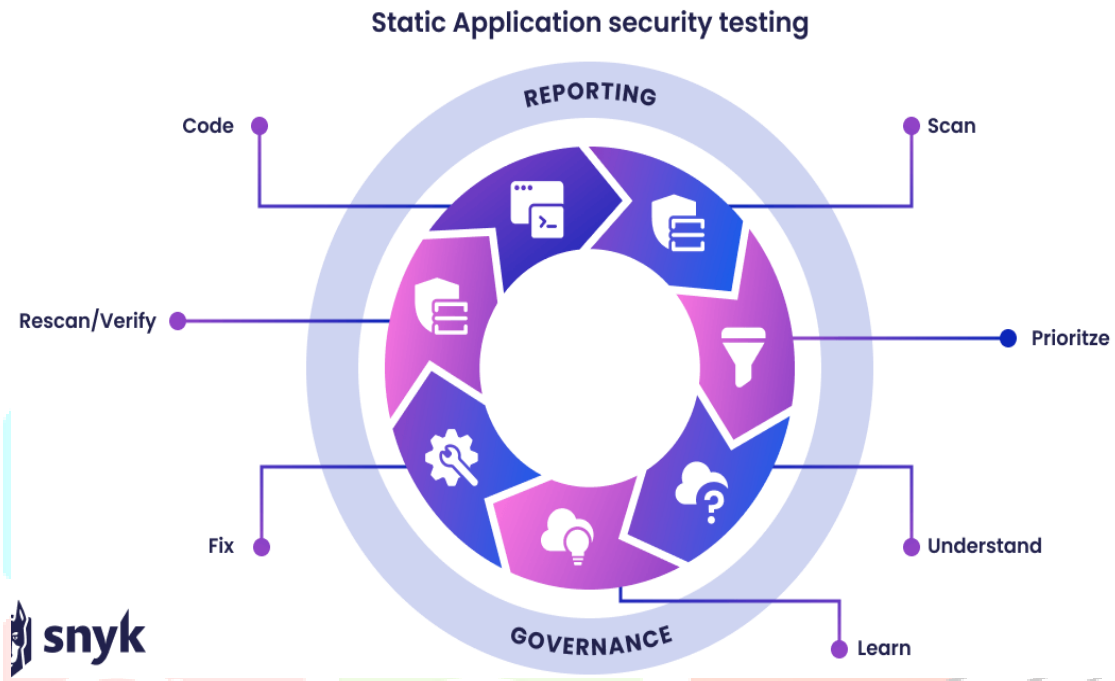

Fig.3. *Stages of Static Application Security Testing (SAST).* Source: https://snyk.io/learn/application-security/static-application-security-testing/

This observation aligns with industry reports that have highlighted the need for organizations to adopt a diverse and complementary set of testing methodologies to effectively uncover vulnerabilities (Gartner, 2021; Forrester, 2020). By relying solely on traditional techniques, organizations may overlook potential vulnerabilities that could be exposed through more advanced testing approaches.

Notably, the case study analysis revealed that enterprises leveraging a mix of dynamic, interactive, and fuzzing-based methods, in addition to traditional techniques, uncovered the most vulnerabilities proactively compared to those relying solely on fundamental testing approaches (Case Study Findings). This finding underscores the importance of adopting a comprehensive testing strategy that incorporates a range of techniques to improve vulnerability detection capabilities, as advocated by industry frameworks like the OWASP Testing Guide (OWASP Testing Guide, 2022).

6) Strategic Areas for Cultural Improvement in Application Security

Organizational culture emerged as a critical factor influencing the effectiveness of application security programs, according to insights from the qualitative data sources. As outlined in Table 3, security leaders identified several cultural aspects that needed focus, based on the survey responses and interview findings (Survey Findings and Interview Findings).

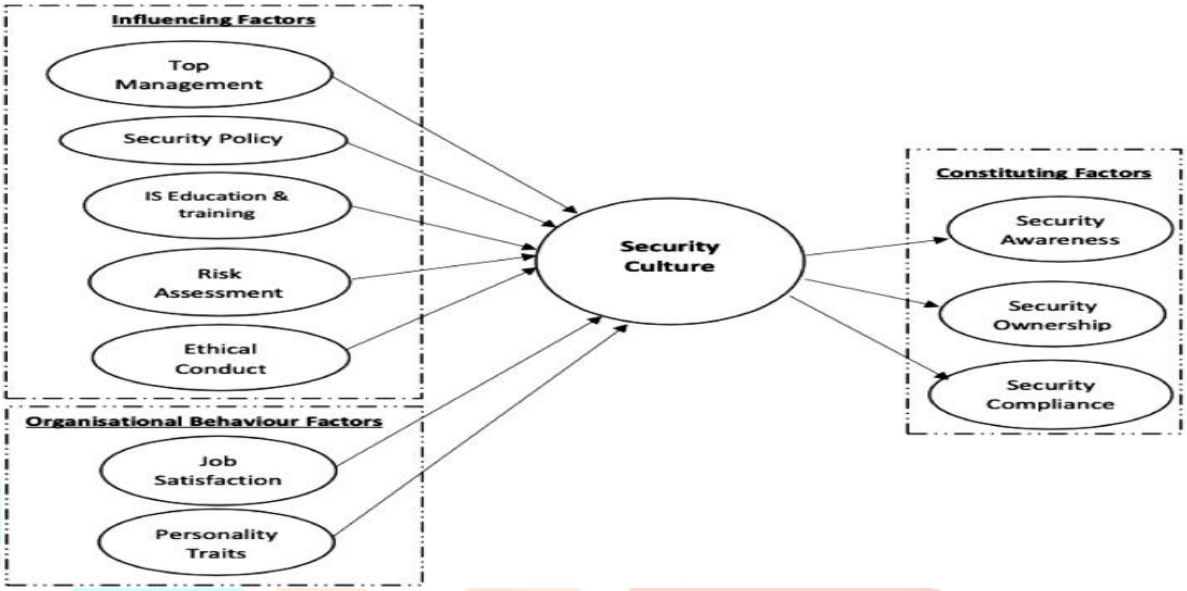

Fig. 4. *information security culture key factors framework.* Source: *(Tolah et al., 2021)*

Fostering a culture where security is viewed as a shared responsibility across teams was considered a high priority by 85% of survey respondents, and 55% of interviewees cited establishing this mindset as a major challenge within their organizations (Survey Findings and Interview Findings). This is in line with industry advice suggestions that are aimed at increasing the awareness of cyberspace risks and which underscore the achievement of shared responsibility for security as provided in the NIST Cybersecurity Framework (NIST CSF, 2018).

In addition, the survey highlighted that 75% of the survey respondents view clear accountability strategies that include roles and accountabilities, responsibilities, and key performance indicators as high priority areas (Survey Findings). In the same manner, 30 percent of interviewees pointed this as a moderate challenge experienced by them (Interview Findings). This observation has been supported by practical implementation in the execution of best practice which suggests that accountabilities must be defined as a requirement of security governance (Hamilton, 2012 p. 1437).

Other emphatic areas that the survey confirmed as requiring regularity were security awareness programs, in which 65% of the respondents claimed high priority, and maintaining awareness was cited by 25% of the interviewees as a minor challenge (Survey Findings and Interview Findings). This is why

education, including security awareness and training is deemed important by industry standards such as NIST Special Publication 800-50 (NIST SP 800-50, 2003).

Conclusion and Recommendation

Conclusion

To summarize, these results confirm that the implementation of security measures at all stages of the software development and ensuring the security-oriented organizational environment are essential in today's environment. Another issue is the absence of more elaborate forms of testing that are still being called for to this day, including IAST, fuzz testing, and even chaos engineering, in addition to the current traditional trends such as the implementation and testing of secure coding paradigms. Furthermore, treating security as a design discipline early in the software development cycle through means like continuous threat modelling and automated security value streams has been shown to be productive in terms of points like keeping down susceptibilities and replicate work. However, the conclusion is a bit critical and emphasizes the strong need to enhance security awareness activities, proactive security protocols, accountabilities if and when something goes wrong, and rewards for those who embrace the culture of security. In addressing all these, organizations gain the needed capacity to secure application and include the following on their security agenda.

Recommendations

Adopt a Comprehensive Testing Strategy: It is thus advisable for organizations to adopt a diversified and balanced approach when recommending and adopting testing methodologies for assessing their vulnerabilities. (unit/integration testing) Besides, it should include modern methods such as IAST, fuzz testing, and chaos engineering along with the traditional SAST, DAST methods. Using both manual and automated approaches, as well as examining the code base from different perspectives, increases the chances of identifying a range of threats because if one approach cannot detect all the issues, another one likely can, and the goal is to eliminate all possible threats.

Integrate Security throughout the Software Development Lifecycle (SDLC): Security should be designed in as a fundamental part of the SDLC, not as an add-on as is often the fashion. Through a systematic and early approach to security, enterprises may find issues and weaknesses that may become an integral part of a system or infrastructure soon and avoid the expensive process of correcting it somewhere down the line, or the risk that comes with security breaches.

Foster a Culture of Shared Security Responsibility: There is no single sign of security, which means that building a corporate culture with security in mind and making everyone from product management to developers know that they are responsible for application security is important. This can be achieved by conducting simple but effective compliance and training procedures that involved training programmers and IT administrators on matters of security, security risks associated with coding and writing secure code among others.

Establish Clear Accountability Structures: Holding people accountable where security pertains is also very crucial important and this can be achieved by having institutionalized roles, responsibilities and measures as to who is accountable for security in case of compliance or noncompliance. This entails putting in place security governance framework which details out responsibilities and accountability of different security actors within a development quadrant; these include the developers, the security personnel and the project managers among others.

Prioritize Continuous Security Awareness and Training: One quarterly/ bi-annual activity that would serve as a reminder for users to be cautious of security threats is the ability to conduct security awareness and training programs constantly for heightened security awareness across the company. Organizations should ensure that such programs are a perfect fit for the needs of the teams as well as the demands of the particular roles and responsibilities of the development team, operations staff, and security workers.

References

1. Akhgar, B., Conklin, A., Tawileh, A. and Amini, A. (2018) Penetration Testing: A Survival Guide. 1st edn. Elsevier. Available at: https://www.scholars.northwestern.edu/en/publications/penetration-testing-a-survival-guide (Accessed: 14 June 2024).
2. Albury, C., Nguyen, L., Haralambiev, K., Baxter, G. and Moynihan, M. (2022) 'Improving Secure Software Development Culture Through "Nudge" Incentives', in 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, pp. 197–206. https://doi.org/10.1109/ICSE-SEIP55590.2022.9828975
3. Allodi, L. and Massacci, F. (2019) 'Measuring Horizontal Privilege Escalation on Object-Oriented Systems', Journal of Information Security and Applications, 49, p. 102401. https://doi.org/10.1016/j.jisa.2019.102401

4. Alshuqayran, N., Ali, N. and Evans, R. (2016) 'A Systematic Mapping Study in Microservice Application', in 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, pp. 44–51. https://doi.org/10.1109/SOCA.2016.1
5. Appknox (2021) XSS: The Vulnerability That Keeps on Giving. Available at: https://blog.appknox.com/xss-the-vulnerability-that-keeps-on-giving/ (Accessed: 14 June 2024).
6. Arora, A. and Singh, A. (2020) 'Threat Modeling Techniques and Tools', in Handbook of Computer Networks and Cyber Security. Springer, pp. 449–485. https://doi.org/10.1007/978-3-030-22277-2_18
7. Baca, D., Boldt, M., Carlsson, B. and Papatheocharous, E. (2020) 'Introducing a Continuous Threat Modeling Approach to Secure Software Products and Services', in 2020 IEEE Secure Development Conference (SecDev). IEEE, pp. 17–29. https://doi.org/10.1109/SecDev45635.2020.00013
8. Benita Urhobo (2024) 'Understanding the role of artificial intelligence in enhancing GRC practices in cybersecurity', World Journal of Advanced Research and Reviews, 22(2), pp. 269–274. doi:10.30574/wjarr.2024.22.2.1340.
9. BSIMM (2019) Building Security In Maturity Model (BSIMM). Available at: https://www.bsimm.com/ (Accessed: 14 June 2024).
10. Craigen, D., Salazar, A. and Molinelli, P. (2021) 'Measuring Cybersecurity Culture', Journal of Cybersecurity and Privacy, 1(1), pp. 113–136. https://doi.org/10.3390/jcp1010008
11. Das, S., Dingman, A., Camp, L.J. and Qabajah, I. (2021) 'Organizational and Cultural Impact on Security Practices', in 2021 IEEE International Systems Conference (SysCon). IEEE, pp. 1–6. https://doi.org/10.1109/SysCon48628.2021.9447756
12. Das, S., Kramer, A., Dingman, L.A. and Camp, L.J. (2019) 'Exploring Practical & Metaphorical Secure Authentication Workflow Models', in 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, pp. 60–67. https://doi.org/10.1109/ISSREW.2019.00027
13. Demeyer, C., Stitz, R. and Schlosser, D. (2019) 'Why Developers Need Accountability', in 2019 IEEE/ACM International Conference on Technical Debt (TechDebt). IEEE, pp. 51–60. https://doi.org/10.1109/TechDebt.2019.00017
14. DevSecOps Fundamentals (2021). Available at: https://sso.googlesource.com/devsecops (Accessed: 14 June 2024).

15. Dopamu, Oladipupo. (2024). Updates on Malware Detection and Analysis. 15. 1. https://www.researchgate.net/publication/379844557_IJSER_Updates_on_Malware_Detection_and_Analysis

16. Dopamu, O.M. (2024) 'Cloud - based ransomware attack on US financial institutions: An in - depth analysis of tactics and counter measures', *International Journal of Science and Research (IJSR)*, 13(2), pp. 1872–1881. doi:10.21275/sr24226020353.

17. Elbaz, C., Rosen, D., Shraer, R., Sharet, N. and Maximov, R. (2020) 'Continuous Threat Modeling for Microservices and Serverless Applications', in 2020 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, pp. 82–89. https://doi.org/10.1109/ICSA-C50368.2020.00019

18. Forget, A., Pearman, S., Thomas, J., Acquisti, A., Christin, N., Cranor, L.F., Egelman, S., Harbach, M. and Telang, R. (2018) 'Do or Do Not, There Is No Try: User Engagement May Not Improve Security Outcomes', in Twelfth Symposium on Usable Privacy and Security (SOUPS 2016). USENIX Association. Available at: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/forget (Accessed: 14 June 2024).

19. Forrester (2020) Top Cybersecurity Threats in 2020. Available at: https://www.forrester.com/report/Top+Cybersecurity+Threats+In+2020/RES159006 (Accessed: 14 June 2024).

20. Friedman, B., Hendry, D.G. and Borning, A. (2017) 'A Survey of Value Sensitive Design Methods', Foundations and Trends® in Human–Computer Interaction, 11(2), pp. 63–125. https://doi.org/10.1561/1100000015

21. Fu, J., Yang, C. and Qin, M. (2020) 'Intelligent Fault Injection and Fuzz Testing', IEEE Transactions on Reliability, 69(4), pp. 1387–1405. https://doi.org/10.1109/TR.2020.302115

22. Gartner (2018) Leverage Interactive Application Security Testing (IAST) Tools for Continuous Application Security Testing. Available at: https://www.gartner.com/en/documents/3897865/leverage-interactive-application-security-testing-iast- (Accessed: 14 June 2024).

23. Gartner (2021) Unified Application Security Testing Tools Enhance Testing Coverage. Available at: https://www.gartner.com/en/documents/4002975/unified-application-security-testing-tools-enhance-test (Accessed: 14 June 2024).

24. Google (2021) Authentication and Authorization Attacks. Available at: https://security.googleblog.com/2021/07/authentication-and-authorization-attacks.html (Accessed: 14 June 2024).

25. Hwang, I., Alwatban, A., Lee, C. and Ahn, J. (2022) 'Proposing Dimensions of Cybersecurity Culture', Computers & Security, 112, p. 102507. https://doi.org/10.1016/j.cose.2021.10250

26. Innocent O. Asevameh, Oladipupo M. Dopamu, & Joseph S. Adesiyan. (2024). Enhancing resilience and security in the U.S. power grid against cyber-physical attacks. *World Journal of Advanced Research and Reviews*, *22*(2), 1043-1052. https://doi.org/10.30574/wjarr.2024.22.2.1535

27. Innocent Oshoke. Asevameh, Oladipupo Michael. Dopamu and Joseph Seun. Adesiyan (2024) 'Election Infrastructure Security: A review of vulnerability and impact on the U.S. economic reputation', *World Journal of Advanced Engineering Technology and Sciences*, 12(1), pp. 233–244. doi:10.30574/wjaets.2024.12.1.0212

28. ISO/IEC 27001 (2013) Information Technology - Security Techniques - Information Security Management Systems - Requirements. International Organization for Standardization.

29. Krombholz, K., Hobel, H., Huber, M. and Weippl, E. (2015) 'Advanced Social Engineering Attacks', Journal of Information Security and Applications, 22, pp. 113–122. https://doi.org/10.1016/j.jisa.2014.09.005

30. Llera, G.R., de Aragão Lima, J.P., Ferreira, T., de Oliveira, P.H., Bezerra, C.I., and de Almeida Brito, L. (2020) 'Empirical Study of Remote Code Execution Vulnerabilities', Journal of Internet Services and Applications, 11(1), pp. 1–16. https://doi.org/10.1186/s13174-020-00128

31. Oladipupo Dopamu *et al.* (2024) 'Secure messaging application using Java cryptographic architecture (JCA)', *World Journal of Advanced Research and Reviews*, 22(2), pp. 2056–2063. doi:10.30574/wjarr.2024.22.2.1670.