International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2017): 7.296

Dynamic Application Security Testing for Payment Applications: A Comprehensive Guide

Pavan Kumar Joshi

Fiserv, USA

Abstract: ***Dynamic Application Security Testing (DAST) plays a crucial role in identifying vulnerabilities in payment applications during their operational phase. As digital payment platforms evolve, security has become an ever-increasing priority, given the rising complexity of cyber threats. While payment applications provide essential services to users in the financial sector, their widespread accessibility makes them prime targets for cyberattacks. This paper explores the security challenges faced by payment applications and highlights the significance of DAST as an essential method for detecting and mitigating these vulnerabilities. The paper presents a comprehensive review of DAST methodologies, including application mapping, security scanning, vulnerability detection, and exploitation analysis, which help developers enhance the security of payment systems. Specific security risks, such as injection attacks, cross-site scripting (XSS), Insecure Direct Object References (IDOR), and misconfigurations, are examined to illustrate how DAST tools effectively detect these threats. Furthermore, the paper provides an in-depth evaluation of the most widely used DAST tools, analyzing their functionality and effectiveness in safeguarding financial data. By emphasizing the importance of integrating security testing into the development life cycle of payment applications, this paper aims to minimize customer risks and reinforce trust in digital payment systems. Ultimately, this study contributes to improving the overall security, reliability, and trustworthiness of payment applications, ensuring safer and more secure transactions for users.***

Keywords: Dynamic Application Security Testing (DAST), payment application security, cybersecurity, Cross-Site Scripting (XSS), SQL injection. information security, application development life cycle, application vulnerabilities remediation

1. Introduction

This is why Dynamic Application Security Testing (DAST) is so useful because it identifies problems during the dynamic use of web applications. This testing is conducted within the test and operation phases of the Software Development Life Cycle (SDLC) of web applications [1]. On the other hand, it's not easy to trace every step of how web apps execute. Additionally, it checks for security vulnerabilities, such as Cross-Site Scripting (XSS), that might be caused by input values using a specified attack string [2]. Therefore, security flaws in online applications that use filters to limit user input are difficult to discover. The creators of IAST (Interactive Analysis Security Testing) set out to address these issues [3]. When Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST) work together, they form IAST, which improves the quality of security tests by overcoming the limitations of individual security analyses [4][5].

Online marketplace apps rely on payment as a core element. Many Fintech businesses have introduced various forms of digital money, such as e-money, which functions as an alternative payment option, along with the advancement of technology. These days, the majority of the steps involved in processing online payments, including those involving e-money, are handled by the Payment Gateway. In the realm of creating new forms of payment systems, IT breakthroughs have become increasingly important [6]. To that purpose, cutting-edge services like online banking and electronic payments have recently emerged. The Internet has provided a new delivery method for banks to reach their customers, and Internet banking has become the most popular form of electronic banking in India. The traditional banking system is giving way to electronic-based business models, and nearly all banks are re-evaluating their approaches to customer relationship management and business process architecture. Particularly in the wake of demonetization in India, the digital payment system has gained tremendous traction [7].

The motivation for this paper stems from the increasing reliance on digital payment applications, which have become essential in modern financial transactions. As these applications facilitate seamless and convenient monetary exchanges, they also attract malicious actors seeking to exploit security vulnerabilities. Given the sensitive nature of financial data involved, ensuring robust security in payment applications is paramount. This paper aims to highlight the importance of implementing DAST in the development and maintenance of payment applications to enhance security, protect user data, and foster trust in digital payment systems. The research is intended to guide financial institutions and developers in adopting best practices for securing payment applications against evolving cyber threats.

A. Organization of the paper

The paper is structured as follows: Section II covers the overview of payment applications. Section III Details the fundamentals of dynamic application security testing. Section IV provides the DAST Vulnerabilities In Payment Applications; Section V discusses the various tools and techniques in DAST for payment applications. Then Section VI presents a Literature review, and identifies research gaps, and VII offers conclusions and future work.

2. Overview of Payment Application

Payment applications, also known as payment apps, are software programs that allow users to make payments from their mobile devices or computers. They can be used to make peer-to-peer payments, pay bills, or pay businesses for products [8]. A "digital payment" is one in which no actual currency, such as cash or checks, are exchanged but rather a variety of internet-based payment methods. Digital payment

methods are becoming increasingly popular due to their ease of use, convenience, and the fact that they allow customers to make payments from any location at any time. This makes them a great alternative to more traditional payment methods and drastically reduces the time it takes for transactions to be completed [9].

1) **PayPal:** PayPal Holdings, Inc. is an American multinational financial technology company operating an online payments system in most countries that support online money transfers; it serves as an electronic alternative to traditional paper methods such as checks and money orders. The company operates as a payment processor for online vendors, auction sites and many other commercial users, for which it charges a fee. [10].

2) **Google Pay:** One form of digital wallet and online payment system created by Google is Google Pay, which goes by several names: G Pay, Pay with Google, and others. When compared to other digital payment systems, Google Pay has a stellar reputation for safety. Google would stop at nothing to ensure the safety of its customers' data stored in the cloud.

3) **MobiKwik:** As a digital wallet and mobile payment system, MobiKwik is another app developed by an Indian company. App creators Bipin Singh and Upasana Taku launched MobiKwik in 2009. Loans, insurance (including life, accident, and fire policies), and mutual funds are all part of Mobikwik's financial offerings.

2.1 Advantage of payment application

There are some advantages of payment applications in the digital way [11].

- **Customers Comfort:** Small business owners can make it easier for customers to pay by accepting mobile payments. Customers can use their cellphones to pay with ease, rather than using credit cards, cash, or checks.
- **Ease of use:** Pay with only one click, no need to enter sensitive information like credit card numbers or passwords. You can also pay instantly by linking your debit or credit card to your bank account. Instantaneous currency conversion and transfers are now a reality.
- **Accessibility:** Everyone should have access to a pricing device if they want to send or receive payments. Access to cellular pricing gadgets can be had whenever and wherever it's needed.
- **Safety and Reliability:** To ensure that payments reach their intended recipients on schedule, encryption (cease-customers) of a charging device might be available at the predicted moment. The gadget is user-friendly, therefore there will be no future losses from fraudulently gathered facts as a result of using it. While device contributors offering guarantees of various kinds might help with some of these issues, the most basic solution is to redesign the device.

3. Fundamentals of Dynamic Application Security Testing (Dast)

Security testing that is executed on an application while it is operating is known as dynamic application security testing (DAST) [12], as opposed to testing static code. Discovering and cataloguing security flaws and misconfigurations is the objective of dynamic application security testing [13]. This security testing methodology and the tools that implement it are both encompassed by the name DAST. Although there is no hard and fast rule regarding which apps or tools can undergo dynamic application security testing, there are two things that are typically true for the methodology and the technologies that employ it:

- The apps that have been tested are web apps. Due to the multiplicity of historical application user interfaces, no known tools have been developed to apply DAST to legacy desktop apps, although in theory it may be possible. The availability of DAST tools for mobile applications has recently expanded.
- Automated processes are an integral aspect of DAST solutions' design. Manually performing dynamic security testing would fall under the purview of penetration testing.

Variable application security testing is also known as outside-in testing, vulnerability scanning, or black-box testing in the application security (AppSec) community [14]. Figure 1 shows the Dynamic Application Security Test Processes.

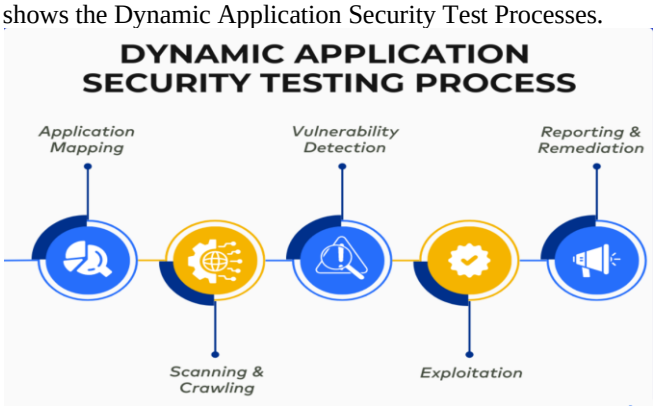

Figure 1: Dynamic Application Security Test Processes

Dynamic Application Security Testing (DAST) is a method for finding vulnerabilities in web applications while they're running. DAST tools simulate automated attacks on an application to identify potential security flaws[2].

2.2 Application Mapping

Applications are made up of a variety of elements—web servers, applications servers, database servers, middleware, network, and storage, to name just a few—that work together and depend on each other to operate. Application mapping is the process of discovering and mapping the interdependencies between applications and all the supporting elements to help

businesses understand how their applications, services, resources, etc. work together and depend on each other to provide the expected output or functionality[15].

a) Security and Crawling

Scan the security of a website, web-based program, network, or file system for vulnerabilities or undesired file changes. This process goes by many names, but one common definition is vulnerability scanning. Application scanning cannot be performed without crawling. By indexing and traversing the site's many pages and states, it facilitates exploration and provides data for testing, which in turn yields findings [16].

b) Vulnerability Detection

In the realm of smart contract security, vulnerability identification is a highly active area of research. As one of the best defenses against contract attacks, it has widespread recognition. The three most popular traditional detection approaches are symbolic execution, formal verification, and fuzz testing [17].

c) Exploitation

In its noun form, "exploit" refers to a piece of code or an entire program that sneaks into a system and starts a denial-of-service (DoS) attack or installs malware like viruses, worms, Trojan horses, adware, or ransomware by exploiting security holes in the system [18].

d) Reporting and Remediation

Reporting and remediation are processes that can help identify and address issues, and then visualize and communicate the results:

- **Remediation:** Involves identifying and fixing issues, such as errors in data, red flags, or cyber threats. For example, data remediation can involve removing duplicate records, standardizing data types, or filling in missing values. In cybersecurity, remediation can involve identifying and mitigating threats to a network or business.
- **Reporting:** Involves visualizing data to highlight trends, patterns, or outliers. Reports can help identify areas for improvement and gather data to make informed decisions. For example, reports can highlight vulnerabilities, issues, or non-compliance with standards.

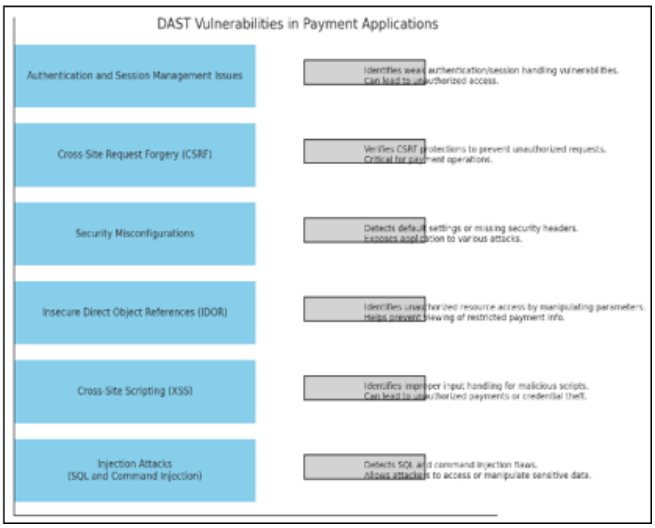

Figure 2: DAST vulnerabilities in payment applications

4. Dast Vulnerabilities in Payment Applications

DAST can help detect various vulnerabilities in payment applications that could be exploited by attackers, as shown in Figure 2. Here's a detailed overview of the main vulnerabilities it can identify:

a) Injection Attacks (SQL and Command Injection)

Finds SQL or command injection flaws that could allow attackers to access or manipulate sensitive data[19].

- **SQL Injection:** DAST finds weaknesses in input fields that allow an attacker to insert malicious SQL queries. This is especially risky for payment applications since it gives hackers access to private client information and allows them to alter records or get past verification.
- **Command Injection:** When user input is sent to the system shell or another command execution environment via the application, this happens. DAST tools search for incorrect input validation, which can provide an attacker access to the server to run arbitrary commands.

b) Cross-Site Scripting (XSS)

Identifies inappropriate input handling that could allow malicious scripts to be injected into the program by attackers.

- **Stored, Reflected, and DOM-based XSS:** DAST finds cases where attackers are able to inject harmful scripts because user inputs are not properly sanitized. These scripts may take over the victim's session, steal their cookies, or execute their every command. Due to the potential for XSS attacks to result in fraudulent payments or the theft of client passwords, payment applications are especially at risk.
- **Sanitization Testing:** DAST checks to make sure user-generated content, URLs, and input fields are properly cleaned to avoid script injection.

c) Insecure Direct Object References (IDOR)

Finds vulnerabilities where users can change parameters to access resources without authorization.

- **Unauthorized Access Testing:** An instance of IDOR occurs when an application secretly makes accessible an internal object reference, such as a database key or file. DAST is useful for finding cases when malicious actors can examine another user's financial information or transaction details by directly manipulating URLs or parameters to access restricted resources.
- **Access Control Verification:** To make sure no unauthorized people may access sensitive information, it verifies that adequate access control measures are in place.

d) Security Misconfigurations

Finds vulnerabilities in the application that could be exploited, such as default settings or missing security headers.

- **Default Configurations:** DAST assists in identifying situations in which the payment application may be susceptible due to idle pages, unpatched software versions, or default accounts. Allow attacks to be made against the application.
- **Missing Security Headers:** To prevent attacks like clickjacking, cross-site scripting, or MIME-sniffing, security headers are essential. DAST finds potentially vulnerable applications by identifying incorrectly set

headers, such as Content-Security-Policy or X-Frame-Options.
- **Debugging Information:** Additionally, DAST looks for accessible debugging information, which can provide hackers access to important data.

e) Cross-Site Request Forgery (CSRF)

Makes sure that CSRF defenses are in place to stop attackers from requesting things that aren't authorized on behalf of users.
- **CSRF Token Testing:** DAST checks to see if CSRF safeguards are present on payment pages. CSRF attacks can be especially dangerous in payment applications since they can lead to unapproved money transfers.
- **Testing Sensitive Actions:** It makes sure that all crucial actions—like entering payment information or modifying account details—are appropriately safeguarded using CSRF tokens, which stop hackers from deceiving users into taking unwanted actions.

f) Authentication and Session Management Issues

Finds weaknesses in session management or insufficient authentication that could allow unwanted access.
- **Broken Authentication:** If an application's authentication mechanisms—such as weak passwords, ineffective password reset procedures, or exposed login endpoints—are susceptible, DAST can assist in determining this.
- **Session Hijacking:** DAST verifies whether the session management is robust—testing for secure cookie attributes like HttpOnly and Secure, and checking if sessions are invalidated correctly upon logout.

5. Various Tools and Techniques in Dast for Payment Applications

Dynamic Application Security Testing (DAST) is a crucial part of the application security lifecycle, especially for payment applications where sensitive data is processed. Table 1 shows the various tools and techniques along with their features and applications for DAST in payment applications.

Table 1: Various Tools and Techniques in DAST For Payment Applications

Tool/Technique	Description	Key Features	Use Case
OWASP ZAP[20]	A web application security scanner that is open-source and can help uncover those pesky vulnerabilities.	Automated scanners, spidering, and intercepting proxy	Testing web applications for OWASP Top Ten vulnerabilities.
Burp Suite[21]	A comprehensive platform for web application security testing with a variety of tools for different testing phases.	Proxy, scanner, repeater, intruder, and extensions	Manual and automated testing of payment applications.
Acunetix[22]	A web application vulnerability scanner that identifies vulnerabilities like SQL Injection and XSS.	Automated scanning, compliance reporting, and scheduling	Scanning e-commerce websites for vulnerabilities.
Netsparker[23]	An automated web application security scanner that identifies vulnerabilities in web applications.	Proof-based scanning, easy integration with CI/CD	Continuous security testing in DevOps environments.
Qualys Web Application Scanner[22]	A cloud-based scanner that provides vulnerability management for web applications.	Continuous monitoring, detailed reporting	Regular assessments of payment gateways and APIs.
AppScan[22]	A DAST tool from IBM that analyses web applications for security vulnerabilities.	Integrates with SDLC, supports various compliance standards	Scanning for vulnerabilities in banking applications.
Snyk[22]	A security tool focusing on open-source and container security, with capabilities for DAST.	Continuous monitoring, automatic fixes for vulnerabilities	Assessing security for applications using third-party libraries.
Cenzic Hailstorm[22]	A web application security testing tool designed for dynamic testing.	Integration with development and security tools	Automated testing during the development phase of payment applications.
W3af[24]	A framework for auditing and attacking online applications that is open-source and helps uncover vulnerabilities.	Plugin architecture, extensive attack capabilities	Testing web applications for various security issues.
HTTP/HTTPS Fuzzing[25]	A method for finding security flaws in online applications by flooding them with random data.	Customisable payloads, testing for error messages	Finding input validation issues in payment applications.

6. Literature Review

In this section, provide some previous work on dynamic application security testing for payment applications.

In [26], due to the growing complexity of online systems, security testing is essential for ensuring that these systems meet all necessary security standards. In this study, they use model-based active testing to tackle this issue. Using IF formalism, they initially describe the behavior of the Web system. Secondly, they use particular algorithms to incorporate security rules modelled in the Nomad language into this IF model. Then, they use a specialized tool that we developed in our lab, HJ2If, to generate tests automatically. As a last step in proving our framework's validity, they provide a quick overview of a travel agency system.

In [27], This section outlines a sequential method for doing search-based application security testing. In the first step, potential candidate tests that could result in good security tests are searched for in the input space. The second step is to narrow the search to a manageable number of options, and then use those candidates to choose and parameterize specific search methods. Based on exploratory security testing, this method begins with evaluating tests for their relation to one another and their capacity to elicit vulnerability symptoms.

One possible way to test the performance of this tiered strategy is by using web application vulnerability scanners as a baseline.

In [28], following static analysis and penetration testing, the study established a standard method for testing the mobile client, transmission, client application, deployment environment, and background application security. This study presents a comprehensive and effective mobile application security risk testing procedure. Pre-line safety review technical advice.

In [29], suggest a web application security evaluation model and state its defining characteristics using the Analytic Hierarchy Process (AHP). They assess the impact of a vulnerability test on the IPB BBS application using the evaluation method suggested in this research. The relationship between the number of vulnerabilities discovered in the security test and the evaluation value calculated by the evaluation function is positively correlated, according to the experimental results. It demonstrates the practicality and reliability of the security evaluation method suggested by this study.

In [30] provides a framework to let analysts test Android app security to address this difficulty. App stores are changing how people get software. This strategy allows for fast acquisition, introduction, maintenance, and enhancement of consumer software. However, this paradigm change has created new security issues. Rapidly assessing market applications' security and robustness is the biggest challenge. The tool suite automatically develops and performs several test scenarios and reports security vulnerabilities to the analyst for an application.

In [31], Continuous software-as-a-service (SaaS) security certification checks if an application meets security standards regularly and automatically. Since SaaS apps leverage web application technology, web application testing looks appropriate for security checks. These methods generally use discrete security tests, which are manually triggered and assessed by humans. They explain significant obstacles and give experimental test results of utilizing SQL Map to continually test for SQL injection vulnerabilities.

In [32], large standardization initiatives and attempts to establish compatible ecosystems and business models use Near Field Communication technology. In this regard, the Global Platform consortium released specifications to allow various actors to handle card material anonymously and adapt the platform to the UICC in Mobile Profile. Experiment, test, and validate these standards. VACAMS (Validation of Content Application Systems) seeks this. Otter Card Systems, Trusted Labs, and GREYC will contribute smart card engineering, testing methodology, and secure electronic transaction knowledge. Developing a UICC Secure Element that meets global Platform criteria and profile and loading and personalizing a payment application "over the air" following mobile device issuance will be the experiment. It will take place on ENSICAEN's experimental electronic payment infrastructure.

In [33], Smartphones, PCs, PDAs, and other devices use NFC for contactless data transmission. Since Bluetooth and it are short-range communication technologies, they are similar. NFC lets consumers pay using their phones. They believe cloud computing can solve many NFC application management challenges, and in this paper, they present our strategy to simplify NFC ecosystem management. They evaluated several SE architectures and discussed our preferred model.

The following Table 2 provides the Comparative analysis of related work for DAST in Payment applications.

Table 2: Comparative analysis of related work for DAST in Payment applications

Reference	Study Focus	Methodology	Tools/ Frameworks	Application Domain	Key Contributions	Limitations
[26]	Security testing of web systems	Model-based active testing, IF formalism, Nomad language	HJ2If	Web applications (Travel agency system)	Introduced a model-based framework for integrating security rules and automated test generation	Limited to specific modelling languages; not tested on payment apps
[27]	Search-based application security testing	Staged approach for test generation and selection	Not specified	Web applications	Developed a staged security testing approach for generating and refining security tests	Lack of specific tools mentioned; exploratory testing focus
[28]	Mobile application security testing	Static analysis, penetration testing, and security mechanisms	Common test technique for mobile apps	Mobile applications	Proposed a comprehensive mobile application security risk testing framework	Focused on mobile apps, lacks emphasis on payment-specific applications
[29]	Security evaluation for web applications	Security evaluation using Analytic Hierarchy Process (AHP)	Not specified	Web applications (BBS application)	Created a security evaluation model correlating vulnerabilities with testing effectiveness	Limited application scope; not directly related to payment systems
[30]	Android app security testing	Automated test development and execution for Android apps	Custom tool suite	Android applications	Developed a framework to automatically test Android app security	Limited to Android apps; manual testing remains costly
[31]	Continuous security certification for SaaS	Continuous security checks, SQL Map for vulnerability testing	SQL Map	SaaS applications	Developed methods for continuous security certification using automated web app testing	Needs improvement for continuous testing adaptability

[32]	NFC payment application security	Security validation for UICC-based NFC systems	VACAMS, UICC Secure Element	NFC payment systems	Experimented with secure NFC payment infrastructure and content management	Focus on standardization; not directly applicable to payment apps
[33]	NFC application management security	Evaluated SE architectures, suggested NFC cloud-based model	Not specified	NFC applications	Proposed a secure and manageable NFC application ecosystem	Focused on architecture rather than DAST for payment security

7. Conclusion and Future Work

A kind of testing, Dynamic Application Security Testing (DAST), has now become indispensable when developing payment applications as more transactions shift online and become easier targets for hackers. Payment applications are very important in facilitating the efficient provision of digital payment services but are open to various cyber-related risks. DAST has emerged as a crucial process needed to protect payment applications from risk situations that may result in severe financial and image loss. The consequences of cyber threats will continue to increase, hence calling for increased security measures like DAST. By conducting application mapping, proper vulnerability scanning, and identifying the problem and solving them, their security level will be improved and user data will be better protected. In this way, the various tools and techniques described in the current guide allow developers and security teams to perform efficient dynamic testing that not only helps to find vulnerabilities but also increases the overall security of applications against modern threats. Lastly, the application of DAST into SDLC is critical for enhancing consumer trust as well as compliance with the regulatory framework in the dynamic environment on digital payments.

Future developments as part of DAST should include (i) AI and machine learning for better detection, (ii) improved API security testing, (iii) inclusion of continuous monitoring in CI/CD processes and (iv) the use of preparedness for post-quantum cryptography. Furthermore, integration of DAST into Zero Trust Architecture (ZTA) could provide better security to the payment application. These innovations shall enhance protection against new threats while at the same time satisfying the laid down regulations.

References

[1] M. D. Ernst, “Static and dynamic analysis: synergy and duality,” *WODA 2003 ICSE Work. Dyn. Anal.*, 2003.

[2] J. Im, J. Yoon, and M. Jin, “Interaction platform for improving detection capability of dynamic application security testing,” in *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, 2017. doi: 10.5220/0006437104740479.

[3] [3] K. Ooms *et al.*, “Combining user logging with eye tracking for interactive and dynamic applications,” *Behav. Res. Methods*, 2014, doi: 10.3758/s13428-014-0542-3.

[4] [4] S. B. Vishwa V Kumar, Salik R Yadav, Frank W Liou, “A digital interface for the part designers and the fixture designers for a reconfigurable assembly system,” *Math. Probl. Eng.*, no. 1, 2013, [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1155/2013/943702

[5] [5] M. T. VISHWA VIJAY Kumar, MUKUL Tripathi, SATISH KUMAR Tyagi, SK Shukla, “An integrated real-time optimisation approach (IRTO) for physical programming based redundancy allocation problem,” *Proc. 3rd Int. Conf. Reliab. Saf. Eng. Udaipur, Rajasthan, India*, pp. 692–704, 2007.

[6] [6] Z. Bezhovski, “The Future of the Mobile Payment as Electronic Payment System,” *Eur. J. Bus. Manag.*, 2016.

[7] [7] P. Pukkasenunk and S. Sukkasem, “An efficient of secure mobile phone application for multiple bill payments,” in *Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016*, 2016. doi: 10.1109/WAINA.2016.63.

[8] [8] S. B. R. Kumar, S. A. Rabara, and J. R. Martin, “A system model and protocol for mobile payment consortia system,” in *Proceedings of the International Symposium on Test and Measurement*, 2009. doi: 10.1109/ICTM.2009.5413011.

[9] [9] M. Massoth and T. Bingel, “Performance of different mobile payment service concepts compared with a NFC-based solution,” in *Proceedings of the 2009 4th International Conference on Internet and Web Applications and Services, ICIW 2009*, 2009. doi: 10.1109/ICIW.2009.112.

[10] Wikipedia.com https://en.wikipedia.org/wiki/PayPal.

[11] X. Li, W. Zhu, and M. He, “Secure remote mobile payment architecture and application,” in *3CA 2010 - 2010 International Symposium on Computer, Communication, Control and Automation*, 2010. doi: 10.1109/3CA.2010.5533752.

[12] A. Dissanayaka, U. D. Annakkage, B. Jayasekara, and B. Bagen, “Risk-based dynamic security assessment,” *IEEE Trans. Power Syst.*, 2011, doi: 10.1109/TPWRS.2010.2089809.

[13] S. G. Priya Pathak, Akansha Shrivastava, “A survey on various security issues in delay tolerant networks,” *J Adv Shell Program.*, vol. 2, no. 2, pp. 12–18, 2015.

[14] J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, “AppSec,” *ACM SIGPLAN Not.*, 2015, doi: 10.1145/2817817.2731199.

[15] P. K. Sahu and S. Chattopadhyay, “A survey on application mapping strategies for network-on-chip design,” *Journal of Systems Architecture*. 2013. doi: 10.1016/j.sysarc.2012.10.004.

[16] G. Jain and Y. Mehandiratta, “Chronicle security against covert crawling,” in *ACM International Conference Proceeding Series*, 2012. doi: 10.1145/2490428.2490434.

[17] P. Liu, J. Su, and X. Yang, “Research on software security vulnerability detection technology,” in *Proceedings of 2011 International Conference on*

Computer Science and Network Technology, ICCSNT 2011, 2011. doi: 10.1109/ICCSNT.2011.6182335.

[18] J. L. Bronstein, "The exploitation of mutualisms," *Ecol. Lett.*, 2001, doi: 10.1046/j.1461-0248.2001.00218.x.

[19] S. Goel, K. Gupta, M. Garg, and A. K. Madan, "Ethical Hacking and Its Countermeasures," *Int. J. Adv. Res. Innov.*, 2014, doi: 10.51976/ijari.231408.

[20] D. Guaman, F. Guaman, D. Jaramillo, and M. Sucunuta, "Implementation of techniques and OWASP security recommendations to avoid SQL and XSS attacks using J2EE and WS-Security," 2017. doi: 10.23919/cisti.2017.7975981.Testing application security with aspects,"

[21] Z. Panczel and C. Walker, "Burp Suite(up) with fancy scanning mechanisms," *SANS Inst.*, 2015.

[22] L. Suto, "Analyzing the Accuracy and Time Costs of Web Application Security Scanners," *San Fr. Febr.*, 2010.

[23] C. Joshi and U. Kumar, "Security Testing and Assessment of Vulnerability Scanners in Quest of Current Information Security Landscape," *Int. J. Comput. Appl.*, 2016, doi: 10.5120/ijca2016910563.

[24] A. Barinas López, A. C. Alarcón Aldana, and M. Callejas Cuervo, "Vulnerabilidad de Ambientes Virtuales de Aprendizaje utilizando SQLMap, RIPS, W3AF y Nessus [Vulnerability in Virtual Learning Environments using SQLMap, RIPS, W3AF and Nessus]," *Vent. Inform.*, 2014, doi: 10.30554/ventanainform.30.276.2014.

[25] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "NetworkProfiler: Towards automatic fingerprinting of Android apps," in *Proceedings - IEEE INFOCOM*, 2013. doi: 10.1109/INFCOM.2013.6566868.

[26] M. Jain and D. Gopalani, "in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 3161–3165. doi: 10.1109/ICEEOT.2016.7755285.

[27] S. Türpe, "Search-Based Application Security Testing: Towards a Structured Search Space," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011, pp. 198–201. doi: 10.1109/ICSTW.2011.96.

[28] J. Guo, H. Jiang, and C. Zhou, "Research on risk analysis and security testing technology of mobile application in power system," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2017, pp. 1–6. doi: 10.1109/EI2.2017.8245498.

[29] D. Jing-Nong and L. Yan-Sheng, "An Effect Evaluation Model for Vulnerability Testing of Web Application," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2010, pp. 382–385. doi: 10.1109/NSWCTC.2010.94.

[30] S. Malek, N. Esfahani, T. Kacem, R. Mahmood, N. Mirzaei, and A. Stavrou, "A Framework for Automated Security Testing of Android Applications on the Cloud," in *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, 2012, pp. 35–36. doi: 10.1109/SERE-C.2012.39.

[31] P. Stephanow and K. Khajehmoogahi, "Towards Continuous Security Certification of Software-as-a-Service Applications Using Web Application Testing Techniques," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, 2017, pp. 931–938. doi: 10.1109/AINA.2017.107.

[32] V. Alimi and M. Pasquet, "Post-Distribution Provisioning and Personalization of a Payment Application on a UICC-Based Secure Element," in *2009 International Conference on Availability, Reliability and Security*, 2009, pp. 701–705. doi: 10.1109/ARES.2009.98.

[33] P. Pourghomi and G. Ghinea, "Managing NFC payment applications through cloud computing," in *2012 International Conference for Internet Technology and Secured Transactions*, 2012, pp. 772–777.