SSRF Safe Route Mapper

Amelia Binte Marzuki, Shafeena Sherin, Khairunnurrin Zurain Binte Khairon Jazan, Lee Ting How Justin, Sudipta Kanti Biswas
Infocomm Technology Cluster
Singapore Institute of Technology
Singapore
{2302159, 2302177, 2302235, 2302275, 2303435}@sit.singaporetech.edu.sg

I. Introduction

The modern web relies on interconnected applications, microservices, and cloud infrastructures that exchange large amounts of data. While this drives innovation, it also creates significant security challenges. One such threat is Server-Side Request Forgery (SSRF), where attackers manipulate a server to make unauthorized requests. These attacks can lead to sensitive data exposure, unauthorized system access, and even full network compromise.

Recent SSRF incidents in major platforms like AWS and Google Cloud highlight the severity of this vulnerability, especially in microservice-heavy environments. Additionally, a recent SSRF vulnerability in PyTorch's Torch-Serve stemming from an arbitrary code execution flaw demonstrates the urgency of addressing this issue. Despite its severity, existing tools often fall short, focusing only on detecting SSRF vulnerabilities without offering actionable insights to address root causes. These limitations reveal an opportunity for a more comprehensive solution.

To bridge this gap, we present the SSRF Safe Route Mapper, an innovative tool designed to revolutionize how SSRF vulnerabilities are detected, visualized, and mitigated. It provides an intuitive solution that:

1. Visualizes request routes, highlighting risky paths within and outside the system.
2. Offers practical remediation guidance to help developers secure their applications.
3. Simulates SSRF exploitation across various protocols, giving security teams a deeper understanding of vulnerabilities.

The SSRF Safe Route Mapper is not just another detection tool but a proactive defense mechanism. It equips developers and security teams with tools and insights to effectively combat SSRF vulnerabilities, empowering them to build more secure web applications. By integrating advanced analysis and actionable fixes, the solution demonstrates technical competency and feasibility within the project's timeline.

II. Background Research

A. *Literature Review*

The detection and mitigation of SSRF vulnerability has been the focus of numerous studies due to its growing prevalence and rising threats of SSRF in web applications. It has been included in the 2021 OWASP Top 10 and MITRE CWE Top 25 most dangerous vulnerabilities. According to a study by Zhang et al. [1], they introduced SSRFuzz, a novel methodology designed to systematically detect SSRF vulnerabilities in PHP web applications. This approach employs techniques such as Dynamic Taint Analysis where the tool employs it to trace user inputs that interact with sensitive server-side request functions. Thus, it helps to reduce potential attack vectors by identifying only inputs capable of triggering SSRF vulnerabilities. Also, fuzzing techniques such as adaptive payload testing were utilized where the tool generates and injects SSRF payloads into input points to test for potential vulnerabilities. The research also highlights the limitations of adopting the techniques such as not being able to scale well for more complex or less common configurations and being limited to URL-based input formats only. By building upon the advanced techniques and methodologies demonstrated in the research mentioned, our tool aims to address its limitations to not only detect the vulnerability but also to integrate actionable fixes such as providing specific recommendations to remediate the identified SSRF vulnerability.

There remain challenges in addressing SSRF vulnerabilities in web applications. Despite their critical impact, SSRF defenses remain underdeveloped and inconsistently applied. According to a study by Wessels et al. [2], there were significant gaps revealed in SSRF defenses from conducting a large-scale static analysis of 27,078 PHP projects. The findings include identifying 237 out of 1,040 applications that utilized SSR sinks having user input-controlled data flows into these sinks as well as over half of the applications were found to have a lack of SSRF defense making them vulnerable to exploit. Secondly, many applications applied common defenses such as URL validation or inadequate deny-listing making them vulnerable to attacks as they were incomplete or able to be bypassed due to improper configurations. Only two applications demonstrated robust allow-list based defenses which include proper URL parsing and input validation, but these approaches may be prone to misconfigurations. Furthermore, it was highlighted that there was a lack of awareness among developers regarding SSRF sinks where existing PHP frameworks and HTTP libraries rarely integrated effective SSRF defenses. Apart from that, a key contribution of the study is the development of the SURFER tool which adopts static analysis to identify SSRF vulnerabilities which helps to trace user inputs to SSR sinks. This approach offers a defensive mechanism to detect potential risks in PHP applications. By building on the gaps and recommendations identified in this research, our tool can incorporate static and dynamic analysis capabilities to identify and mitigate SSRF vulnerabilities to improve the security posture of the application.

Server-side vulnerabilities like Cross-Site Request Forgery (CSRF) and Server-Side Request Forgery (SSRF) exploit trust relationships in web applications. This leads to unauthorized actions or data breaches. A study on CSRF by Dr.M.Buvana highlights the persistent challenges posed by such vulnerabilities which emphasizes the importance of multi-layered defense mechanisms to mitigate risks. Techniques such as CSRF tokens, referrer header validation, and same-site cookies are discussed as effective yet limited strategies. The study stresses that there is no single solution that can fully address these threats, advocating for a robust combination of token-based verification, enhanced session management, and real-time monitoring to prevent exploitation [3]. These principles resonate with the challenges of SSRF mitigation, where trust relationships between internal systems are often abused. By adopting a similar multi-layered approach, SSRF vulnerabilities can be better understood and mitigated.

Further emphasizing the need to address security vulnerabilities in web applications, a systematic mapping study by S. Rafique et al. analyzed empirical research on vulnerability detection approaches. It highlights the significant growth in reported web application vulnerabilities, such as SQL injection, cross-site scripting (XSS), and CSRF, over the past two decades. It also emphasizes that many of these vulnerabilities stem from flaws in software design, coding, and implementation, particularly at the application layer. According to the findings, 75% of security attacks occur at this layer, underscoring the critical need for robust detection and mitigation strategies. By mapping existing solutions against the software development lifecycle and the OWASP Top 10, the study identified gaps in detection and proposed future directions. Notably, vulnerabilities like SSRF often arise from poor design and inadequate input validation, validating the approach of the SSRF Safe Route Mapper, which seeks to address these issues during implementation while providing actionable insights to improve design practices [4].

A study on "Preventing Server-Side Request Forgery Attacks" examines strategies for mitigating SSRF attacks, particularly in cloud and internal network environments where these attacks exploit internal server trust. These vulnerabilities can lead to data breaches and unauthorized use of server resources. The study proposes a defense mechanism that uses a proxy server to handle external resource requests. This helper server is isolated from the internal network by strict firewall rules to enhance security. The research categorizes various types of SSRF attacks (such as in-band and out-of-band) and the consequences they can cause, including credential theft. It also addresses potential bypass methods, like DNS rebinding. The study evaluates the proposed solution's effectiveness by testing it on OWASP Vulnerable Web Applications, showing that it offers a minimal performance impact while effectively mitigating SSRF risks [5].

The article "Systematic Study of SSRF" by Christian Rossow et al. provides a comprehensive exploration of Server-Side Request Forgery (SSRF) attacks, offering valuable insights that align with our project's objectives. It highlights how SSRF exploits servers' trust in internal and external requests, enabling attackers to perform actions like port scanning, data exfiltration, and privilege escalation. The study introduces a classification framework for SSRF attacks and presents a novel tool to detect vulnerabilities. It emphasizes the need for robust mitigation strategies, such as input validation and network segmentation, and underscores the importance of visualization for understanding SSRF pathways [6]. These findings reinforce the necessity of tools like SSRF Safe Route Mapper, which prioritize mapping and remediating request vulnerabilities while simulating real-world attack scenarios to strengthen security awareness.

B. *Existing Tools and Solutions*

Currently, several tools are available to identify and mitigate Server-Side Request Forgery (SSRF) vulnerabilities in web applications. One of the solutions is SSRFMap released by SlowMist. It is an automatic SSRF fuzzer and exploitation tool that takes a Burp request file as input and fuzzes parameters to identify and exploit SSRF vulnerabilities [7]. It provides support for various modules such as remote code execution (RCE) through FastCGI, Redis, Github Enterprise (versions < 2.8.7), and Zaddix, MySQL command execution, SMTP mail sending, port scanning, network scanning, and file operations such as reading /etc/passwd. However, SSRFMap mainly focuses on detection and exploitation, without providing features for remediation guidance or request route visualization.

The second tool currently available is the Amazon Web Services (AWS) Web Application Firewall (WAF) which would be able to define and enforce specific rules that filter out malicious traffic before it reaches their applications. AWS WAF allows users to define and enforce specific rules to filter out malicious traffic before it reaches their applications. It includes key features such as rule-based filtering, where users can create custom rules to block requests targeting internal IP addresses or using unsafe protocols like file:// or gopher://, which are commonly exploited in SSRF attacks [8]. AWS WAF also offers managed rule groups that provide pre-configured rules for common threats, helping to mitigate vulnerabilities and address application security requirements. Additionally, AWS WAF integrates with AWS Shield for enhanced protection against DDoS attacks and other threats. While AWS WAF is effective at blocking known malicious patterns, it has limitations in addressing advanced SSRF attack techniques such as DNS rebinding, which can bypass traditional IP-based filtering [9].

Netsparker, developed by Invicti Security, is also another tool that is designed to detect SSRF vulnerabilities by signature-based and behavior-based detection methods across APIs, and web services. Firstly, Netsparker has the ability to confirm the exploitability of identified vulnerabilities, minimizing false positives and equipping security teams with actionable insights. Its Proof-Based Scanning feature validates that reported vulnerabilities are genuine and exploitable, thus streamlining the remediation process and conserving time and resources [10]. With that ability to validate the reported vulnerabilities, it is also complemented by its ability in providing comprehensive reports and remediation guidance, which are especially beneficial for development teams working on web applications [11]. However, there are few shortcomings for Netsparker. Firstly, it has issues detecting unknown SSRF vulnerabilities. It is able to detect known SSRF patterns but struggles with more complex or novel SSRF techniques that rely on application-specific logic or unique configurations. Secondly, Netsparker does not have the ability for context-based vulnerability detection [12]. For example, an application might have custom input validation rules, specific user roles, or complex workflows that dictate how data is processed and accessed. Netsparker, being an automated tool, relies on predefined patterns and signatures to detect vulnerabilities [13]. While this works well for common vulnerabilities like SSRF, SQL injection, or XSS, it may fail to identify issues that arise from application-specific logic or custom configurations. Without an understanding of the application's context, Netsparker might overlook these risks, leaving the application exposed to sophisticated attacks. This is particularly concerning for organizations with complex, custom-built applications where context-aware security is critical.

III. Proposed Solution

The SSRF Safe Route Mapper is designed to identify, analyze, and mitigate SSRF vulnerabilities in web applications. Its primary goal is to detect vulnerabilities while providing actionable guidance to help developers proactively address these issues, improving web application security.

The tool operates in three phases:

1. **Input Analysis:** Users input a URL or test case representing real-world applications or simulated attacks. The tool traces the request's path, validating it against known SSRF exploit patterns like accessing internal IPs, unsafe protocols, or bypassing access controls.
2. **Risk Visualization:** The tool generates an interactive map of the request's flow, highlighting critical vulnerabilities such as unauthorized access to private networks.
3. **Mitigation Guidance:** Tailored recommendations include restricting access to network ranges, sanitizing user inputs, and implementing whitelists and firewalls. Step-by-step guidance, code snippets, and configuration examples facilitate quick remediation.

What Sets It Apart: Unlike traditional SSRF detection tools, the SSRF Safe Route Mapper emphasizes prevention and education. Its scenario-based approach simulates attack scenarios across protocols and configurations, ensuring teams can understand specific risks their applications face. By integrating advanced visualization and tailored guidance, the tool fosters a deeper understanding of SSRF vulnerabilities and solutions.

Design and Implementation: The tool will use Flask (Python) for backend processing, including input validation and request simulation, and D3.js for interactive visualizations. Azure Virtual Machines will simulate vulnerable web servers, enabling realistic testing environments. This approach demonstrates technical competency while ensuring feasibility within the project's 8-week timeline. The tool will also feature a knowledge base of SSRF mitigation techniques, updated based on industry best practices.

Testing: The tool's effectiveness will be validated through scenario simulations, focusing on common SSRF attack vectors like internal IP discovery and protocol abuse. Benchmarks against tools like Burp Suite will highlight its unique focus on visualization and mitigation. User feedback will drive iterative improvements, ensuring usability and clarity.

By combining detection, visualization, and actionable guidance, the SSRF Safe Route Mapper empowers developers to address SSRF vulnerabilities effectively. This comprehensive approach not only strengthens application security but also demonstrates innovation and feasibility, aligning with the project's goals.

REFERENCES

[1] Y. Zhang, X.Wang, and M. Yang, "Where URLs Become Weapons: Automated Discovery of SSRF Vulnerabilities in Web Applications," 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, [Online]. Available: https://ieeexplore-ieee-org.singaporetech.remotexs.co/document/10646755 (accessed Jan 17, 2025)

[2] M. Wessels, S. Koch, G. Pellegrino, and M. Johns, "SSRF vs. Developers: Study of SSRF-Defenses in PHP Applications," in Proceedings of the 33rd USENIX Security Symposium, Philadelphia, PA, USA, Aug. 2024. [Online]. Available. https://www.usenix.org/conference/usenixsecurity24/presentation/wessels (accessed Jan 17, 2025)

[3] B. M., "Mitigating cross-site request forgery vulnerabilities: Evaluating current strategies and proposing defense mechanisms," [Online] Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4783867 (accessed Jan 17, 2025)

[4] S. Rafique et al., Web application security vulnerabilities detection approaches: A systematic mapping study, [Online} Available: https://ieeexplore.ieee.org/document/7176244 (accessed Jan. 17, 2025)

[5] B. Jabiyev, E. Kirda, A. Kharraz, and O. Mirzaei, Preventing server-side request forgery attacks, [Online] Available: https://seclab.nu/static/publications/sac21-prevent-ssrf.pdf (accessed Jan. 17, 2025)

[6] G. Pellegrino, O. Catakoglu, D. Balzarotti, and C. Rossow, Uses and abuses of server-side requests - christian rossow, https://christian-rossow.de/publications/ssr-raid2016.pdf (accessed Jan. 17, 2025).

[7] Swissky, "swisskyrepo/SSRFmap," *GitHub*, Jun. 14, 2024. [Online] Available: https://github.com/swisskyrepo/SSRFmap (accessed Jan. 19, 2025)

[8] AWS Managed Rules for AWS WAF - AWS WAF, AWS Firewall Manager, and AWS Shield Advanced, *Amazon.com*, 2025. [Online] Available: https://docs.aws.amazon.com/waf/latest/developerguide/aws-managed-rule-groups.html (accessed Jan. 19, 2025).

[9] Behind the Firewall: Bypassing AWS WAF's SSRF Managed Rules, *Medium.com*, 2025. [Online] Available: https://medium.com/%40bcksec/behind-the-firewall-bypassing-aws-wafs-ssrf-managed-rules-3671a789c815 (accessed Jan. 19, 2025).

[10] Avoid False Positives with Proof-Based Scanning | Invicti," Invicti, Mar. 11, 2022 [Online] Available: https://www.invicti.com/features/proof-based-scanning/ (accessed Jan. 19, 2025)

[11] [Invicti Security, "New Invicti Research Reveals Proof-Based Scanning Automatically Confirms 94% of Direct-Impact Vulnerabilities with 99.98% Accuracy," *Prnewswire.com*, Sep. 28, 2021[Online] Available: https://www.prnewswire.com/news-releases/new-invicti-research-reveals-proof-based-scanning-automatically-confirms-94-of-direct-impact-vulnerabilities-with-99-98-accuracy-301385889.html (accessed Jan. 19, 2025)

[12] What needs improvement with Netsparker Web Application Security Scanner?, *PeerSpot*, Jul. 29, 2021. [Online] Available: https://www.peerspot.com/questions/what-needs-improvement-with-netsparker-web-application-security-scanner (accessed Jan. 19, 2025)

[13] H. N. Security, "Review: Netsparker Enterprise web application scanner," *Help Net Security*, Oct. 19, 2020. [Online] Available: https://www.helpnetsecurity.com/2020/10/19/review-netsparker-enterprise-web-application-scanner/ (accessed Jan. 19, 2025)