STM JOURNALS

Journal of
Operating Systems
Development & Trends

ISSN: 2454-9355
Volume 12, Issue 3, 2025
September–December
DOI (Journal): 10.37591/JoOSDT

https://journals.stmjournals.com/joosdt

Review JoOSDT

Review on STROT: An Intelligent, Automated Red Teaming Framework using Deep Q-Learning

Pratik Suhas Pawar[1,*], Shubham Pandurang Sakhare[1], Vishnu Latish Nair[2], Vishal G. Puranik[2]

Abstract

This review examines STROT, an intelligent and automated red teaming framework designed to enhance penetration testing through the integration of machine learning and system automation. Unlike traditional approaches that require manual coordination between reconnaissance, vulnerability assessment, and exploit execution, STROT unifies these processes within a single architecture composed of a network analyzer, an AI-driven intelligence module, and an autonomous attack engine. At its core, STROT employs Deep Q-Learning to dynamically select optimal exploits based on real-time analysis, allowing the system to adapt its strategies over time. Evaluated in controlled environments, STROT demonstrated improved efficiency, reduced detection risk, and faster privilege escalation compared to conventional tools. This review highlights the framework's strengths in automation, adaptability, and stealth while also discussing its current limitations and future potential for broader cybersecurity applications.

Keywords: Cybersecurity, red teaming, penetration testing, deep Q-learning, exploit automation, network reconnaissance, artificial intelligence, vulnerability assessment, stealth attacks, cyberattack simulation

INTRODUCTION

In today's rapidly evolving digital landscape, cybersecurity threats have grown in both complexity and scale. Organizations are increasingly relying on proactive security measures such as red teaming and penetration testing to assess the resilience of their systems against potential cyber-attacks. Traditional red teaming methods, however, are often time-consuming, manually intensive, and reliant on the expertise of skilled professionals. These methods typically involve the use of multiple disconnected tools for tasks like network reconnaissance, vulnerability scanning, and exploit deployment. This fragmented approach not only slows down the testing process but also increases the likelihood of detection by defensive systems [1].

***Author for Correspondence**
Pratik Suhas Pawar
E-mail: pratikpawarkarad3@gmail.com

[1]Student, Department of Technology, Savitribai Phule Pune University, Pune, Maharashtra, India
[2]Assistant Professor, School of Electronics and Telecommunication Engineering, Savitribai Phule Pune University, Pune, Maharashtra, India

Received Date: October 03, 2025
Accepted Date: October 25, 2025
Published Date: November 19, 2025

Citation: Pratik Suhas Pawar, Shubham Pandurang Sakhare, Vishnu Latish Nair, Vishal G. Puranik. Review on STROT: An Intelligent, Automated Red Teaming Framework using Deep Q-Learning. Journal of Operating Systems Development & Trends. 2025; 12(3): 1–7p.

The reviewed study introduces STROT (Stealthy Tool for Root Oriented Tunneling) as a novel solution to these challenges. STROT is designed as an intelligent, fully automated red teaming framework that integrates reconnaissance, decision-making, and attack execution within a single platform. It leverages advancements in machine learning, particularly deep reinforcement learning, to optimize the exploitation process. By employing Deep Q-Learning, STROT dynamically selects the most effective exploits based on real-time network and system analysis, significantly reducing the need for human intervention [2].

This review explores the architectural components, methodologies, and experimental results presented in the original study, with a focus on evaluating STROT's performance, scalability, and practical relevance in real-world cybersecurity scenarios. Special attention is given to its adaptive learning capabilities, stealth mechanisms, and potential for broader applications in both offensive and defensive security domains. Through this analysis, the review aims to provide insights into the effectiveness and innovation behind STROT, while also identifying opportunities for further development and refinement in the realm of intelligent cybersecurity tools [3].

OVERVIEW OF THE STROT FRAMEWORK

STROT (Stealthy Tool for Root Oriented Tunneling) is designed as an end-to-end automation platform for red teaming operations, combining reconnaissance, vulnerability analysis, and exploit execution in a unified, intelligent framework. The system architecture is modular and comprises four key components: the Command-Line Interface (CLI), Network Analyzer, Intelligence Module, and Attack Engine.

The process begins with user initiation via the CLI, which triggers a sequence of automated actions. The Network Analyzer performs stealth scanning using tools such as Nmap and Scapy to identify open ports, services, operating systems, and version information. The output is passed to the Tokenization module, which parses and encodes the scan results into a format suitable for machine learning-based processing [4].

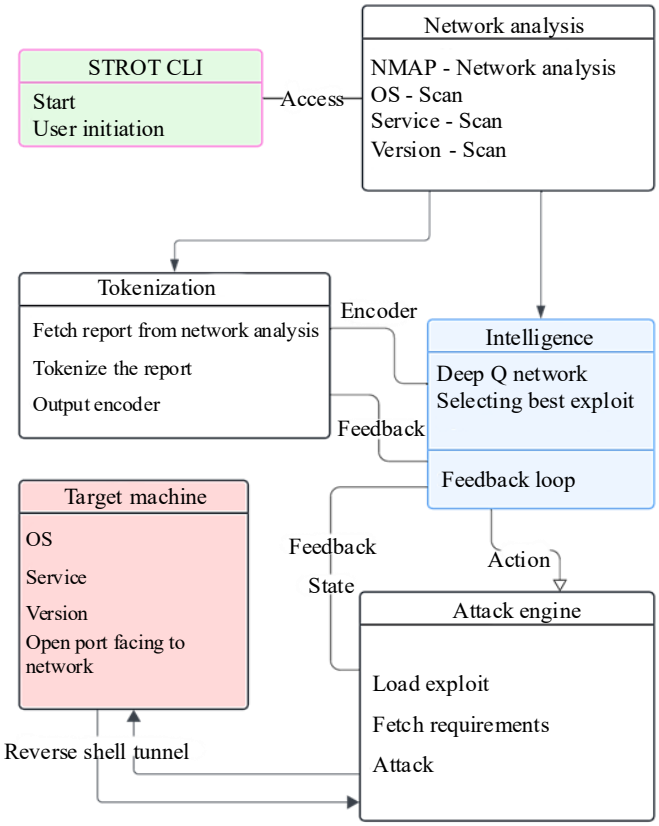

Figure 1. Framework workflow.

This data is then sent to the Intelligence Module, where a Deep Q Network (DQN) analyzes the state of the target system and selects the most suitable exploit. The Attack Engine loads the selected exploit, checks for necessary prerequisites, and executes the attack. A feedback loop updates the DQN based on the success or failure of the attack, allowing STROT to learn and improve its decision-making over time.

Figure 1 illustrates the complete workflow of the STROT framework and the interactions between its components.

This modular pipeline enables STROT to operate autonomously, offering a streamlined and adaptive red teaming tool that minimizes human intervention while maximizing efficiency and stealth [5].

Technical Analysis and Evaluation

The STROT framework distinguishes itself through its use of deep reinforcement learning to automate and optimize the red teaming process. This section provides a detailed technical analysis of its internal components and evaluates the system's performance based on its design, implementation, and experimental results [6].

At the core of STROT is the Deep Q-Learning (DQL) model, which enables intelligent exploit selection by learning optimal strategies through interaction with the environment. The model is trained using a Markov Decision Process (MDP) formulation, where the state space captures characteristics such as system vulnerabilities, operating systems, open ports, and historical exploit outcomes. The action space comprises available exploits, while the reward function provides positive or negative feedback based on the success of each exploitation attempt. This reinforcement learning approach allows STROT to dynamically adapt its attack strategy, improving efficiency over time.

The Network Analyzer performs reconnaissance using stealth-focused techniques. It combines raw packet manipulation (via Scapy), TCP/IP socket programming, and the Nmap Python API to identify target hosts, services, and software versions. This hybrid method allows STROT to operate discreetly, avoiding full TCP handshakes and reducing the chance of detection by intrusion detection systems (IDS) [7].

The Attack Engine executes selected exploits and incorporates a feedback loop to inform future decision-making. If an exploit fails, the result is used to adjust the Q-values in the intelligence module, refining subsequent attack attempts. This closed-loop design is key to the system's learning capability and contributes to its high adaptability.

Experimental results show that STROT significantly improves attack efficiency. In controlled sandbox environments such as Metasploitable, the framework demonstrated faster time-to-root access, fewer exploitation attempts, and lower decision latency as the model trained over time. Episode reward and Q-value graphs confirmed the effectiveness of the learning algorithm, while decreasing latency metrics indicated faster, more confident decision-making [8].

Overall, the technical architecture of STROT successfully integrates machine learning with classic network exploitation workflows, offering a promising foundation for scalable and intelligent offensive cybersecurity tools.

COMPARISON WITH EXISTING TOOLS AND METHODS

Red teaming and penetration testing traditionally rely on a combination of discrete tools, each addressing a specific phase of the attack lifecycle. Commonly used tools include Nmap for network discovery, Metasploit for exploit development and execution, and Nessus for vulnerability scanning. While effective, these tools are often used in isolation, requiring manual correlation of results and decision-making from the user. This disjointed approach increases the complexity, time, and expertise required to execute comprehensive assessments.

STROT (Stealthy Tool for Root Oriented Tunneling) offers a paradigm shift by integrating reconnaissance, intelligence-based decision-making, and exploit execution into a single, automated framework. Its use of deep reinforcement learning enables dynamic exploit selection based on

environmental feedback, which distinguishes it from static or rule-based systems [9, 10].

Table 1 provides a detailed comparison between STROT and three widely used tools: Metasploit, Nmap, and Nessus, across key capabilities and features relevant to red teaming.

Table 1. Comparison of STROT with traditional cybersecurity tools.

Feature/Capability	STROT	Metasploit	Nmap	Nessus
Automation	Full (End-to-End)	Partial	None	Partial
Exploit Selection	AI-based (DQL)	Manual or scripted	Not applicable	Not applicable
Vulnerability Scanning	Integrated	Limited	Port/service only	Comprehensive
Learning Capability	Reinforcement Learning	None	None	Basic heuristics
Stealth Mechanisms	Yes (Scapy, Passive Scan)	No	Basic options	Yes (configurable)
Multi-Stage Attack Handling	Limited (Current)	Yes	No	No
Feedback Loop/ Adaptability	Yes	No	No	Limited
Open Source	Yes	Yes	Yes	No (Commercial)
User Skill Requirement	Low-Moderate	High	Moderate	Moderate

From the comparison, it is evident that STROT offers unique advantages in several areas:

1. *Automation:* Unlike traditional tools, STROT eliminates the need for multiple separate stages by providing full workflow automation, from scanning to exploitation.
2. *AI-Based Decision Making:* STROT leverages Deep Q-Learning to make real-time, context-aware exploit decisions. This is a major advancement over manual or signature-based exploit selection.
3. *Stealth Capabilities:* By using passive scanning and partial TCP handshakes via Scapy, STROT minimizes its detection footprint, which is essential in adversarial environments.
4. *Feedback-Driven Adaptation:* STROT's closed-loop architecture allows it to learn from past actions, adjust its future behavior, and improve its performance over time, something none of the compared tools natively support.
5. *Ease of Use:* Despite incorporating complex machine learning models, STROT is designed with lower operational complexity for the end-user, making it more accessible than tools like Metasploit that demand in-depth expertise.

However, it is also important to acknowledge that STROT is currently limited in areas such as multi-host coordination and zero-day vulnerability handling, which remain strongholds of more mature and specialized platforms. Nevertheless, STROT's intelligent integration of scanning, analysis, and execution presents a novel direction for red team automation.

In conclusion, STROT complements and extends traditional toolsets rather than replacing them outright. It exemplifies a new generation of cybersecurity tools where autonomous decision-making, adaptability, and stealth are built-in capabilities, rather than manually managed layers.

STRENGTHS AND CONTRIBUTIONS

The STROT framework brings several notable strengths and contributions to the field of offensive cybersecurity, particularly in the context of red teaming and automated penetration testing. Its design and implementation reflect a thoughtful integration of machine learning, system automation, and stealth-focused engineering.

1. *End-to-end automation:* One of STROT's primary strengths lies in its fully automated workflow. By integrating reconnaissance, vulnerability analysis, exploit selection, and execution into a single pipeline, STROT significantly reduces the need for manual intervention.

This streamlining enhances efficiency and allows red teamers to focus on strategic planning rather than operational execution.

2. *Intelligent exploit selection via reinforcement learning:* The use of Deep Q-Learning distinguishes STROT from traditional penetration testing tools. By continuously learning from attack outcomes, STROT adapts its behavior over time, leading to more accurate and context-aware exploit strategies. This intelligence results in faster privilege escalation and a reduction in ineffective or redundant exploit attempts.
3. *Stealth and evasion capabilities:* STROT employs a variety of stealth scanning techniques, such as partial TCP handshakes and passive monitoring, to reduce the likelihood of detection by intrusion detection systems. This focus on low-noise operations makes it especially effective in adversarial environments where stealth is critical.
4. *Modular and scalable design:* Its modular architecture allows for easy integration of new scanning techniques, vulnerability databases, or exploit modules. This scalability makes STROT adaptable to different network environments and evolving threat landscapes.
5. *Real-world applicability:* The framework has been successfully tested in practical, sandboxed environments such as Metasploitable, demonstrating its capacity to autonomously identify vulnerabilities and execute exploits with high reliability. These tests validate the system's real-world relevance and usability.
6. *Open source and extendable:* STROT's open-source nature encourages collaboration, review, and continuous improvement. It also promotes transparency in how automated cybersecurity tools operate, making it accessible for educational, research, and professional use.

Collectively, these strengths position STROT as a forward-looking tool that not only enhances red teaming operations but also sets the stage for future innovations in autonomous cybersecurity.

LIMITATIONS AND AREAS FOR IMPROVEMENT

While STROT presents a promising advancement in the field of automated red teaming, it is not without its limitations. Understanding these constraints is essential for contextualizing its current capabilities and identifying avenues for future development.

1. *Dependence on known vulnerabilities:* STROT relies on a predefined exploit database to map detected services to known vulnerabilities. As a result, its effectiveness is limited when encountering zero-day vulnerabilities or highly customized systems. The tool does not yet possess the capability to discover or exploit previously unknown attack vectors autonomously.
2. *Limited multi-host coordination:* Currently, STROT is optimized for targeting individual machines within a network. It lacks the ability to coordinate complex, multi-stage attacks across multiple hosts simultaneously. In real-world scenarios, especially within enterprise networks, advanced red team operations often involve lateral movement, pivoting, and coordinated exploits across interconnected systems, capabilities not yet fully addressed in the framework.
3. *Generalization challenges in dynamic environments:* Although STROT uses reinforcement learning to adapt over time, its training and performance are highly dependent on the environment it is exposed to. It may struggle to generalize its learned strategies when deployed in networks with configurations or architectures significantly different from its training data.
4. *Limited integration with defensive security tools:* STROT is designed primarily as an offensive tool. It currently lacks integration with blue team mechanisms such as deception systems, honeypots, or threat detection platforms. Such integration could enhance its value for full-spectrum cyber simulations and adversary emulation.
5. *Evaluation scope and realism:* Most performance evaluations were conducted in controlled environments using standard testbeds like Metasploitable. While these provide a good starting point, real-world networks often contain a broader mix of technologies, obfuscation, and defensive mechanisms. Further validation in production-like settings is needed to assess robustness under realistic conditions.

6. *Computational resource requirements:* Deep Q-Learning models, particularly during training, can be computationally intensive. Although the trained model executes relatively efficiently, initial training might require specialized hardware or significant time investment, which could be a barrier for some users.

Addressing these limitations will be crucial for evolving STROT from a promising prototype into a mature, widely applicable cybersecurity tool. Enhancements in adaptability, scalability, and cross-functional integration will significantly extend its operational scope.

Future Directions

While STROT represents a significant step forward in automated red teaming, there are several key areas where future development can enhance its capability, adaptability, and real-world applicability. These directions aim to address current limitations and align the tool with evolving cybersecurity demands.

1. *Real-time integration with exploit databases:* To remain current with newly discovered vulnerabilities and exploits, STROT could be enhanced with real-time synchronization from trusted public repositories such as ExploitDB or Metasploit. Automating this update process would improve its ability to detect and exploit the latest known vulnerabilities without manual intervention.
2. *Expansion to multi-target and lateral movement strategies:* Future versions of STROT should support coordinated attacks across multiple hosts, including lateral movement, privilege escalation chains, and pivoting techniques. This would make the tool more effective for emulating advanced persistent threats in complex enterprise environments.
3. *Adaptive evasion and anti-detection techniques:* To further enhance stealth, STROT could incorporate reinforcement learning strategies that adapt in real-time to avoid triggering intrusion detection and prevention systems. This could include noise calibration, dynamic payload generation, or randomized execution paths based on environmental feedback.
4. *Hybrid integration with defensive systems:* There is strong potential to extend STROT's utility by integrating it into defensive simulation environments. This could include coupling with deception platforms, honeynets, or threat-hunting systems for use in adversary emulation, cyber range exercises, or red-blue team collaboration scenarios.
5. *Support for zero-day and Heuristic-based exploitation:* Future iterations could move beyond signature-based exploit selection by incorporating heuristic analysis or generative models capable of identifying and crafting potential zero-day attacks. This would require advanced training datasets and ethical oversight, but could position STROT as a next-generation offensive AI tool.
6. *Graphical user interface (GUI) and usability enhancements:* Improving accessibility through a user-friendly GUI would open STROT to a broader audience, including educators, students, and entry-level professionals. Visualization dashboards for exploit paths, model learning progression, and risk metrics could further enhance usability and interpretability.
7. *Performance optimization and lightweight deployment:* Ongoing work could focus on reducing the computational footprint of the intelligence module, making STROT deployable on low-resource systems or as a portable utility for field penetration tests.

By pursuing these enhancements, STROT can evolve from a proof-of-concept framework into a mature, robust platform suitable for both offensive and defensive cybersecurity operations in real-world, production-grade environments.

CONCLUSION

STROT represents a significant advancement in the field of offensive cybersecurity by introducing an intelligent, automated approach to red teaming. By combining traditional network scanning techniques with deep reinforcement learning, it addresses several longstanding challenges associated with manual penetration testing, such as inefficiency, limited scalability, and high detection risk. Its

modular architecture, comprising network analysis, intelligent exploit selection, and automated attack execution, enables streamlined, end-to-end operation without requiring continuous human intervention.

The framework's integration of Deep Q-Learning marks a notable contribution to the evolving intersection of machine learning and cybersecurity. Through adaptive learning, STROT can refine its attack strategies based on real-time outcomes, allowing it to become more effective and stealthy over time. Evaluation in sandbox environments, such as Metasploitable, confirms its ability to identify vulnerabilities and execute successful exploits quickly and efficiently.

Despite its innovations, STROT also faces limitations in areas such as zero-day detection, multi-host coordination, and generalization to real-world networks. These constraints highlight the importance of continued research and development. Future directions, including the expansion of exploit databases, integration with defensive tools, and support for more complex attack scenarios, present clear opportunities to enhance the framework's impact.

In summary, STROT demonstrates the growing potential of artificial intelligence in cybersecurity offense, shifting the paradigm from static, rule-based tools to intelligent, learning-based systems. As the threat landscape continues to evolve, tools like STROT will play a critical role in helping security professionals stay ahead of adversaries through intelligent automation, adaptive learning, and strategic simulation.

REFERENCES

1. Pawar PS, Sakhare SP, Nair VL, Puranik VG. STROT: stealthy tool for root oriented tunneling - a red teaming tool. Int Res J Eng Technol. 2025 Apr; 12(4): 1–10.
2. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S. Human-level control through deep reinforcement learning. Nature. 2015 Feb; 518(7540): 529–33.
3. Mo K, Ye P, Ren X, Wang S, Li W, Li J. Security and privacy issues in deep reinforcement learning: Threats and countermeasures. ACM Comput Surv. 2024 Feb 23; 56(6): 1–39.
4. Chen Y, Li Y, Lu Y, Pan Z, Ji YC, Chen Y, Li Y, Shen Y. Understanding the security risks of websites using cloud storage for direct user file uploads. IEEE Trans Inf Forensics Secur. 2025 Feb 20; 20: 2677–2692.
5. Lyon GF. (2009 Jan 1). Nmap network scanning: The official Nmap project guide to network discovery and security scanning. [Online]. Insecure.
6. Kennedy D, O'gorman J, Kearns D, Aharoni M. Metasploit: the penetration tester's guide. No starch press; 2011 Jul 15.
7. Geer D, Harthorne J. Penetration testing: a duet. 18th Annual Computer Security Applications Conference, 2002. Proceedings. Las Vegas, NV, USA, 2002. p. 185-95. doi:10.1109/CSAC.2002.1176290.
8. Applebaum A, Miller D, Strom B, Korban C, Wolf R. Intelligent, automated red team emulation. In Proceedings of the 32nd annual conference on computer security applications. 2016 Dec 5; 363–373.
9. Zennaro FM, Erdődi L. Modelling penetration testing with reinforcement learning using capture-the-flag challenges: Trade-offs between model-free learning and a priori knowledge. IET Inf Secur. 2023 May; 17(3): 441–57.
10. MITRE. (2025). MITRE ATT&CK. [Online]. Available from: https://www.mitre.org/focus-areas/cybersecurity/mitre-attack

