

DOI: https://doi.org/10.48009/1_iis_2025_137

Generative AI for web application pentesting

Raul Diaz-Parra, *Universidad de Lima, Lima-Peru, jdiazp@ulima.edu.pe*

Abstract

The integration of Generative AI into cybersecurity practices has opened new possibilities for automating and enhancing offensive security operations. This study explores the application of ShellGPT in the context of web application penetration testing using the OWASP Web Security Testing Guide (WSTG) as the methodological framework. The experiment targeted a vulnerable application and systematically progressed through reconnaissance, enumeration, and exploitation phases. Notably, ShellGPT successfully identified and exploited an SQL injection vulnerability, enabling full data extraction from the backend database. Results show that LLMs can generate accurate commands and support non-expert users throughout the penetration testing lifecycle.

Keywords: Cybersecurity, Offensive security, Penetration testing, Large language models, ShellGPT

Introduction

The financial impact of data breaches continues to rise, the global average cost of a data breach reached USD 4.88 million, 10% increase from the previous year (IBM, 2024). This increase is largely driven by operational disruptions, loss of customers, and post-breach remediation efforts. Therefore, Penetration testing has become a fundamental practice in contemporary cybersecurity due to the increasing frequency, complexity, and impact of cyberattacks affecting organizations worldwide (Alhamed & Rahman, 2023). As digital infrastructures expand and integrate deeply with operational processes across sectors such as healthcare, finance, and government, new vulnerabilities emerge that adversaries can exploit (Lazarov, 2025). Penetration testing allows organizations to proactively simulate real-world attacks in controlled environments, helping to identify vulnerabilities and prioritize remediation efforts (Alhamed & Rahman, 2023). Regulatory frameworks such as ISO 27001, PCI DSS and NIST Cybersecurity Framework have emphasized the need for periodic security assessments to ensure compliance and readiness against evolving threats (PCI Security Standards Council, 2024; National Institute of Standards and Technology, 2024; ISO, 2022). Therefore, systematic penetration testing is necessary to protect organizational assets, maintain stakeholder trust, and enhance incident preparedness.

In addition, the growing use of Large Language Models (LLMs) has significantly transformed the cybersecurity landscape, particularly within the offensive domain such as penetration testing. GPT-4 has demonstrated advanced capabilities in natural language understanding, code generation, and contextual analysis, facilitating their integration into automated penetration testing (PT) scenarios and web application exploitation (Uddin et al., 2025; Chen et al., 2024). This technological evolution has led to combine generative AI with pentesting methodologies from reconnaissance to complex attack simulation (McKinnel et al., 2019).

Recent studies have highlighted the value of LLMs in planning and executing attack paths. (Chen et al., 2024) emphasize that LLMs can be used to simulate attack scenarios, generate offensive actions, and adapt

attacks to the target environment. Similarly, Uddin et al., (2025) use LLM to generate payloads to evaluate the effectiveness of system resilience through simulated attacks. These capabilities not only accelerate the exploitation cycle but also enhance the effectiveness of identifying vulnerabilities and attack vectors.

However, the use of LLMs in offensive contexts also raises significant ethical and technical challenges. McKinnel et al. (2019) point out that although AI integration into vulnerability assessment improves efficiency and scalability, it introduces risks such as false positives, reliance on limited training data, and potential misuse by malicious actors. This study has the purpose of examining how Generative AI (GenAI) tools can be integrated into web application penetration testing and assessing their overall impact on the penetration testing process.

Literature Review

In recent years, the rise of Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs), has transformed how penetration testing is conceived and executed. Several recent studies have demonstrated the potential of these technologies to automate tasks traditionally reserved for human experts, thereby reducing costs and accelerating security assessment cycles (Hilario et al., 2024; Zaydi & Maleh, 2025).

Hilario et al. (2024) used ChatGPT 3.5 in conjunction with ShellGPT to carry out full penetration tests in simulated environments. Their work demonstrated that an LLM can guide the entire pentesting cycle from reconnaissance to exploitation while offering accurate commands and generating quality reports. This study focused mainly on evaluating network services.

By other hand, Martínez et al. (2025) conducted a comparative analysis of ChatGPT-4, Claude Opus, and GitHub Copilot under the Penetration Testing Execution Standard (PTES) framework. They found that while LLMs cannot fully replace human pentesters, they provide significant support, especially during intelligence gathering and exploitation phases. This finding aligns with that of Zaydi & Maleh (2025), who highlight how tools like ShellGPT automate key tasks such as dynamic payload generation and privilege escalation.

Frameworks like CAI (Mayoral-Vilches et al., 2025) and BreachSeek (Alshehri et al., 2024) push this automation further by deploying multi-agent architectures that integrate LLMs with toolchains to simulate full-scale attacks in realistic scenarios. Both frameworks focused on infrastructure assessment. Fang et al. (2024) demonstrated that red teams who used LLM agents can exploit zero-day vulnerabilities without prior knowledge.

Wang & Johnson (2024) proposed an AI-moderated knowledge discovery model that leverages ChatGPT, to analyze simulated network traffic and identify potential vulnerabilities. Their study, grounded in strategic principles from *The Art of War*, demonstrates how GenAI can improve the efficiency of knowledge discovery in pentesting scenarios. However, they also reveal critical limitations, including reduced accuracy with large input files and the need for human validation of AI-generated findings.

Based on the above discussion, this section has identified two gaps in the literature: 1) application of LLMs to perform penetration testing against web application, and 2) lack of studies using penetration testing methodologies on the context of modern web application.

Theoretical Framework

Generative AI

Generative AI (GenAI), particularly through Large Language Models (LLMs), is reshaping penetration testing by enabling automation, contextual reasoning, and intelligent task execution across the attack lifecycle. GenAI tools can simulate adversarial behavior, craft dynamic payloads, and generate commands in natural language, lowering barriers to complex offensive operations while accelerating test cycles (Aung et al., 2025).

The use of ShellGPT in this study is justified by its specialized ability to understand and generate shell language commands, an essential capability in IT operations where precision and efficiency are critical. Unlike general-purpose language models such as GPT-2 (Uddin et al., 2025) or programming-oriented variants like CodeGPT (Lu et al., 2021), ShellGPT is specifically tailored for the shell domain through techniques such as pre-tokenization and Equivalent Command Learning, which enhances the model's semantic understanding of functionally similar commands (Shi et al., 2023). These domain-specific adaptations enable ShellGPT to outperform baseline models in key tasks such as command recommendation, error correction, and natural language-to-shell translation, making it a robust tool for reliable automation and intelligent assistance in operating system environments.

In this context, ShellGPT, a command-line interface that integrates OpenAI's GPT models, operationalizes GenAI for cybersecurity by translating user prompts into executable shell commands, supporting reconnaissance, exploitation, and privilege escalation directly from the terminal (TheR1D, 2023).

Such integration enhances tester productivity and bridges skill gaps by enabling even non-experts to perform sophisticated operations. As noted by Hilario et al. (2024), tools like ShellGPT exemplify how LLMs can augment penetration testing workflows, though they also introduce risks such as hallucinated commands and overreliance on AI outputs, underscoring the need for expert oversight and secure deployment practices.

Web Security Testing Guide (WSTG)

The OWASP Web Security Testing Guide (WSTG) serves as a comprehensive and community-driven methodology for assessing the security of modern web applications. WSTG has become a widely recognized standard in the field of application security testing (OWASP, 2023). It offers a structured and modular approach to identify, verify, and document security vulnerabilities, aligning with both industry best practices and evolving threat models.

WSTG is organized in categories that reflect the various stages and components of a typical web application penetration testing, including Information Gathering, Configuration and Deployment Management Testing, Authentication Testing, Session Management Testing, Access Control Testing, Input Validation Testing, and Business Logic Testing, among others. Each test case is methodically documented with objectives, test procedures, and references to known vulnerability, such as OWASP Top Ten (OWASP, 2023).

WSTG is a mature, adaptable, and theoretically grounded methodology that reflects both the practical needs of penetration testers and the strategic imperatives of secure software engineering. Therefore, this study will use this methodology.

Methodology

The penetration testing experiment was based on using interactions with ChatGPT through the command-line interface (CLI), utilizing sgpt to run shell commands in Kali Virtual Machine. The selected web application target was <http://testaspnet.vulnweb.com>. The methodology used was WSTG to emulate the experience of a pentester who has used security tools manually but does not yet have the syntax or tool parameters memorized.

In this study, we employ a pre-trained GPT model without applying any fine-tuning or additional training. This decision aligns with the methodology proposed by Hilario et al. (2024), who emphasize that “no fine-tuning was conducted on the GPT model; all outputs were generated using the default behavior of the model as provided by OpenAI.” Instead of modifying the model’s internal parameters, we adopt a prompt engineering approach to guide the model’s output through carefully crafted input examples. This strategy enables the evaluation of the model’s zero-shot performance and its inherent capabilities in translating natural language instructions into shell commands without domain-specific adaptation.

Experimentation

Information Gathering

Step 1: Conduct Search Engine Discovery Reconnaissance for Information Leakage (WSTG-INFO-01)

Objective: Map the exposed attack surface of a target domain, revealing subdomains, IP addresses and services.

Terminal Input for GenAI:

```
sgpt --shell "Execute recon of domain vulnweb.com"
```

Terminal Output:

```
whois vulnweb.com && dig vulnweb.com && nslookup vulnweb.com && host vulnweb.com && theHarvester -d vulnweb.com -l 100 -b all
```

Terminal input for GenAI:

```
sgpt --shell "Execute enumeration of vulnweb.com with amass"
```

Terminal Output:

```
amass enum -d vulnweb.com
```

Explanation: To gather preliminary intelligence on the target domain vulnweb.com, a set of standard reconnaissance and enumeration tools were executed. These tools are needed to identify domain registration details, DNS configurations, email addresses, and associated subdomains, which are crucial for mapping the attack surface. By combining passive and active enumeration, the reconnaissance phase was able to construct a detailed profile of the domain's internet-facing assets like <http://testaspnet.vulnweb.com>

Step 2: Fingerprint Web Application Framework (WSTG-INFO-08)

Objective: Fingerprint the components used by the web application such as Web Application Firewall.

Terminal Input for GenAI:

```
sgpt --shell "Identify web application firewall on http://testaspnet.vulnweb.com"
```

Terminal Output:

```
wafw00f http://testaspnet.vulnweb.com
```

Explanation: To determine whether the target web application is protected by a Web Application Firewall (WAF), the tool wafw00f was employed. It is identified Microsoft Web Application Firewall was implemented.

Step 3: Identify Application Entry Points (WSTG-INFO-06)

Objective: Identify possible entry points like ports and services

Terminal Input for GenAI:

```
sgpt --shell "Scan all ports and services of http://testaspnet.vulnweb.com"
```

Terminal Output:

```
nmap -p- -sV testaspnet.vulnweb.com
```

Terminal Input for GenAI:

```
sgpt --shell "Scan ports and services of http://testaspnet.vulnweb.com"
```

Terminal Output:

```
nmap -sV testaspnet.vulnweb.com
```

Explanation: To assess the external attack surface of the target host <http://testaspnet.vulnweb.com>, two port scanning strategies were employed using Nmap. These scans aim to identify open ports, associated services, and potentially vulnerable software versions running on the target host. It was identified that the web server has IIS service on port 80.

Step 4: Enumerate Applications on Webserver (WSTG-INFO-04)

Objective: Enumerate the applications or URLs in the web server.

Terminal Input for GenAI:

```
sgpt --shell "Execute dirsearch recursively of http://testaspnet.vulnweb.com and exclude codes 500,404,403"
```

Terminal Output:

```
dirsearch -u http://testaspnet.vulnweb.com -r -x 500,404,403
```

Explanation: To identify hidden resources within the web application hosted at <http://testaspnet.vulnweb.com>, the tool dirsearch was employed for automated directory enumeration. The command used, enables recursive scanning of directories and subdirectories while excluding common HTTP error responses (500, 404, and 403) to reduce false positives and irrelevant output. This method facilitates the discovery of unlinked or misconfigured endpoints that may expose sensitive files, administrative interfaces, or development artifacts. Recursive enumeration is particularly effective in uncovering deeply nested content structures that could be overlooked during standard scanning. It was identified different URLs within the web server, one of them is <http://testaspnet.vulnweb.com/default.aspx>

Step 5: Map Execution Paths Through Application (WSTG-INFO-07)

Objective: Map the target application and URLs for input validation.

Terminal Input for GenAI:

```
sgpt --shell "Execute list URLs of http://testaspnet.vulnweb.com/default.aspx"
```

Terminal Output:

```
curl -s http://testaspnet.vulnweb.com/default.aspx | grep -oP '(?<=href=")[^"]*'
```

Explanation: This command operates by first using curl -s to silently fetch the raw HTML content of the specified webpage, suppressing any progress or error messages to ensure clean output. The resulting HTML is then piped into grep, configured with the -o flag to output only the matching substrings and the -P flag to enable Perl-Compatible Regular Expressions (PCRE). The regular expression `(?<=href=")[^"]*` utilizes a positive lookbehind assertion to locate content that immediately follows the attribute href=", capturing all non-quote characters until the next closing quote. This precisely isolates the values of href attributes, i.e., the URLs embedded in hyperlinks within the HTML structure. The result is a list of all link targets present in the document, including both internal and external references. So, it was identified the URL <http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html>

Input Validation

Step 6: Testing for SQL Injection (WSTG-INPV-05)

Objective: Identify SQL injection points.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" --batch
```

Explanation: This command leverages sqlmap, an open-source penetration testing tool designed to detect and exploit SQL injection flaws. The -u flag specifies the target URL containing potentially injectable parameters, specifically, id and NewsAd, which are query string inputs. By including both parameters in the full URL, the tool is able to systematically test them for vulnerability to SQL-based attacks. The --batch option enables non-interactive execution by automatically selecting default choices during the testing process, making it suitable for automation or scripted evaluations. This approach eliminates the need for human intervention during runtime and ensures consistent testing conditions. SQLmap identified three successful payloads: Boolean, Stacked and Time-based injection.

Step 6.1: List Databases

Objective: Enumerate databases of the web application.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to enumerate databases of  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" --dbs
```

Explanation: For our analysis, the terminal executed SQLMap command which allowed us to validate the tool's effectiveness in identifying injection points and retrieving schema-level metadata. This process supports the broader goal of demonstrating automated techniques for detecting security weaknesses in web applications. It was identified six databases: master, acuforum, acuservice, model, msdb and tempdb.

Step 6.2: Show current database.

Objective: Show current database of the web application.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to identify current database of  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" --current-db
```

Explanation: To determine the active database context of the target web application, the terminal used the `--current-db` option in SQLMap. This allowed us to identify the specific database currently in use after confirming the presence of a SQL injection vulnerability. It was identified database “acublog”. This database wasn’t identified when list databases.

Step 6.3: List tables.

Objective: Show list of tables of a determined database.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to list tables of acublog of  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" --  
dbms="Microsoft SQL Server" -D acublog --tables
```

Explanation: To enumerate the tables within the target database, the terminal executed SQLMap with the `-tables` option, specifying the database name acublog and the DBMS as Microsoft SQL Server. This allowed us to retrieve the full list of table names, facilitating further analysis and targeted data extraction. Such enumeration is a critical step in assessing the extent of a SQL injection vulnerability and understanding the underlying database structure. It was identified three tables in acublog database: comments, news and users.

Step 6.4: List columns.

Objective: Show list of columns of a determined table.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to list columns of users table of acublog database  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" -D acublog  
-T users --columns
```

Explanation: To analyze the structure of a specific table, the terminal used SQLMap with the `--columns` option targeting the users table within the acublog database. This step enabled the extraction of column names and data types, which is essential for understanding the schema and planning further data retrieval. Such structural enumeration provides critical insights into how sensitive information may be stored and accessed through SQL injection vulnerabilities. It was identified three columns in table users: alevel, uname and upass.

Step 6.5: Dump table.

Objective: Show data of determined table.

Terminal Input for GenAI:

```
sgpt --shell "Execute sqlmap to dump users table of acublog database  
http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html"
```

Terminal Output:

```
sqlmap -u "http://testaspnet.vulnweb.com/ReadNews.aspx?id=0&NewsAd=ads/def.html" -D acublog -  
T users --dump
```

Explanation: To complete the exploitation process, the terminal used SQLMap's --dump option to extract the full contents of the users table from the acublog database. This operation enabled the retrieval of stored user records, which is essential for evaluating the sensitivity of the exposed data. The ability to access such information underscores the critical impact of SQL injection vulnerabilities on data confidentiality and system security. It was dumped data of table "users"

Results and Discussions

The experiment demonstrated that ShellGPT, when integrated with traditional penetration testing tools such as sqlmap, nmap, dirsearch, and amass, can effectively automate multiple stages of the OWASP Web Security Testing Guide (WSTG). Full reconnaissance of the target domain (testaspnet.vulnweb.com) was successfully achieved, revealing subdomains, IP addresses, and identifying an IIS service running on port 80. During enumeration, several URLs and application entry points were discovered. Notably, a SQL injection vulnerability was identified in the id parameter of the ReadNews.aspx endpoint. This allowed for the enumeration of databases, tables, and columns, and ultimately enabled data extraction from the users table. The LLM accurately generated shell commands at each step without requiring the user to memorize tool syntax or parameters, thereby streamlining the testing workflow.

These findings confirm that generative models like ChatGPT, when accessed via ShellGPT, can not only assist but enhance the efficiency of web application penetration testing. In alignment with prior research (Hilario et al., 2024; Uddin et al., 2025), this study reinforces the capability of LLMs to support tasks such as reconnaissance, exploitation, and post-exploitation. Nevertheless, limitations were observed: overreliance on AI-generated commands can lead to ineffective or incorrect execution, as previously noted by Aung et al. (2025). Moreover, while the process was largely automated, human validation remains essential to confirm findings and mitigate false positives. This work also highlights a notable gap in current literature: the lack of empirical studies applying LLMs in structured methodologies like WSTG specifically for modern web applications, thus reinforcing the significance of this contribution.

To assess the reliability and accuracy of ShellGPT in the penetration testing workflow, a comparative analysis was conducted between the commands generated by the language model and those typically expected from experienced human pentesters. The table below summarizes this comparison across each major testing step in alignment with the OWASP Web Security Testing Guide (WSTG).

Table 1: Commands Generated by ShellGPT vs. Expected Commands from a Penetration Tester

WSTG Test Case	Prompt to ShellGPT	Command Generated by ChatGPT	Expected Command for pentester	Difference	Assessment
WSTG-INFO-01	"Execute recon of domain vulnweb.com"	whois vulnweb.com && dig vulnweb.com && nslookup vulnweb.com && host vulnweb.com && theHarvester -d vulnweb.com -l 100 -b all	whois, dig, nslookup, host, theHarvester	None	Correct and comprehensive
WSTG-INFO-01	"Execute enumeration of vulnweb.com with amass"	amass enum -d vulnweb.com	amass enum -d <domain>	None	Fully accurate
WSTG-INFO-08	"Identify web application firewall on http://testaspnet.vulnweb.com"	wafw00f http://testaspnet.vulnweb.com	wafw00f <url>	None	Accurate
WSTG-INFO-06	"Scan all ports and services..."	nmap -p- -sV http://testaspnet.vulnweb.com	nmap -p- -sV <ip/domain>	Minor: URL instead of IP	Functionally equivalent
WSTG-INFO-06	"Scan ports and services..."	nmap testaspnet.vulnweb.com -sV	nmap -sV <domain>	None	Correct
WSTG-INFO-04	"Execute dirsearch recursively..."	dirsearch -u http://testaspnet.vulnweb.com -r -x 500,404,403	dirsearch -u <url> -r -x <codes>	None	Fully correct
WSTG-INFO-07	"Execute list URLs of..."	`curl -s	grep -oP '(?<=href=")[^"]*'	curl with regex-based parsing	None
WSTG-INPV-05	"Execute sqlmap to ..."	sqlmap -u "<url>" --batch	sqlmap -u <url> with options	None	Fully valid
WSTG-INPV-05	"List databases"	sqlmap -u "<url>" --dbs	sqlmap -u <url> --dbs	None	Accurate
WSTG-INPV-05	"Show current database"	sqlmap -u "<url>" --current-db	sqlmap -u <url> --current-db	None	Accurate
WSTG-INPV-05	"List tables"	sqlmap -u "<url>" --dbms="Microsoft SQL Server" -D acublog --tables	With or without --dbms	Optional flag, not required	Slightly verbose but acceptable
WSTG-INPV-05	"List columns"	sqlmap -u "<url>" -D acublog -T users --columns	sqlmap -u <url> -D <db> -T <table> --columns	None	Fully accurate
WSTG-INPV-05	"Dump table"	sqlmap -u "<url>" -D acublog -T users --dump	sqlmap -u <url> -D <db> -T <table> --dump	None	Accurate

Across all test cases, ShellGPT generated commands that fall within the accepted syntax used by pentesters. Minor differences such as including optional flags (e.g., --dbms) or referencing URLs instead of IPs in some tools (e.g., Nmap) did not affect command execution or results. The outputs were syntactically correct and functionally equivalent to what would be expected from experienced penetration testers. This high degree of alignment supports the conclusion that ShellGPT performed reliably and accurately, reinforcing its value in supporting web application penetration testing efforts.

The integration of generative AI in penetration testing raises significant ethical considerations that must be addressed to ensure responsible and secure deployment. While tools like ShellGPT demonstrate the capacity to autonomously generate effective commands for tasks such as vulnerability scanning or SQL injection testing, their use must be bounded by legal and ethical frameworks. Unauthorized or

indiscriminate application of these capabilities may lead to violations of privacy, unauthorized access, and potential damage to target systems, even if unintentionally. According to Shi et al. (2023), the ethical deployment of generative AI in cybersecurity contexts requires robust governance mechanisms, clear consent protocols, and continuous oversight to mitigate risks of misuse or autonomous deviation from intended tasks.

Conclusion

This study demonstrates that Generative AI, particularly through ShellGPT, can significantly enhance the web application penetration testing process by automating critical tasks from reconnaissance to exploitation. Integrating LLMs with structured methodologies such as OWASP WSTG not only accelerates the testing cycle but also lowers the entry barrier for less experienced analysts. However, consistent expert supervision is necessary to mitigate risks from AI misuse. This work lays a foundational step for future research into the reliability, accuracy, and secure deployment of LLMs in real-world offensive security scenarios.

References

- Alhamed, M., & Rahman, M. M. H. (2023). A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions. *Applied Sciences (Switzerland)*, 13(12). <https://doi.org/10.3390/app13126986>
- Alshehri, I., Alshehri, A., Almalki, A., Bamardouf, M., & Akbar, A. (2024). *BreachSeek: A Multi-Agent Automated Penetration Tester*. 1–7. <http://arxiv.org/abs/2409.03789>
- Aung, Y. L., Christian, I., Dong, Y., Ye, X., Chattopadhyay, S., & Zhou, J. (2025). Generative AI for Consumer Internet of Things: Challenges and Opportunities. *IEEE Consumer Electronics Magazine*. <https://doi.org/10.1109/MCE.2025.3532890>
- Chen, Z., Kang, F., Xiong, X., & Shu, H. (2024). A Survey on Penetration Path Planning in Automated Penetration Testing. *Applied Sciences (Switzerland)*, 14(18). <https://doi.org/10.3390/app14188355>
- Fang, R., Bindu, R., Gupta, A., Zhan, Q., & Kang, D. (2024). *Teams of LLM Agents can Exploit Zero-Day Vulnerabilities*. <http://arxiv.org/abs/2406.01637>
- Hilario, E., Azam, S., Sundaram, J., Imran Mohammed, K., & Shanmugam, B. (2024). Generative AI for pentesting: the good, the bad, the ugly. *International Journal of Information Security*, 23(3), 2075–2097. <https://doi.org/10.1007/s10207-024-00835-x>
- IBM. (2024). *Informe Cost of a Data Breach 2024 Resumen ejecutivo*. <https://www.ibm.com/downloads/cas/BPG4KJRX>
- ISO. (2022). *ISO 27001 - Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. <https://www.iso.org/standard/27001>

- Lazarov, W. (2025). Computers & Security Penterep : Comprehensive penetration testing with adaptable interactive checklists. *Computers & Security*, 154(February 2024), 104399. <https://doi.org/10.1016/j.cose.2025.104399>
- Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., ... Liu, S. (2021). CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation. *Advances in Neural Information Processing Systems*.
- Martínez, A. L., Cano, A., & Ruiz-Martínez, A. (2025). *Generative Artificial Intelligence-Supported Pentesting: A Comparison between Claude Opus, GPT-4, and Copilot*. 1–31. <http://arxiv.org/abs/2501.06963>
- Mayoral-Vilches, V., Navarrete-Lozano, L. J., Sanz-Gómez, M., Espejo, L. S., Crespo-Álvarez, M., Oca-Gonzalez, F., Balassone, F., Glera-Picón, A., Ayucar-Carbajo, U., Ruiz-Alcalde, J. A., Rass, S., Pinzger, M., & Gil-Uriarte, E. (2025). *CAI: An Open, Bug Bounty-Ready Cybersecurity AI*. 1–38. <http://arxiv.org/abs/2504.06017>
- McKinnel, D. R., Dargahi, T., Dehghantanha, A., & Choo, K. K. R. (2019). A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. *Computers and Electrical Engineering*, 75, 175–188. <https://doi.org/10.1016/j.compeleceng.2019.02.022>
- National Institute of Standards and Technology. (2024). *The NIST Cybersecurity Framework (CSF) 2.0*. 32. <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf>
- OWASP. (2023). *Web Security Testing Guide*. <https://owasp.org/www-project-web-security-testing-guide/latest/>
- PCI Security Standards Council. (2024). *Payment Card Industry Data Security Standard*. June. [https://docs-prv.pcisecuritystandards.org/PCI DSS/Standard/PCI-DSS-v4_0_1.pdf](https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI-DSS-v4_0_1.pdf)
- Shi, J., Jiang, S., Xu, B., Liang, J., Xiao, Y., & Wang, W. (2023). ShellGPT: Generative Pre-trained Transformer Model for Shell Language Understanding. *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 671–682. <https://doi.org/10.1109/ISSRE59848.2023.00082>
- Shi, X., Evans, R., & Shan, W. (2022). Solver engagement in online crowdsourcing communities: The roles of perceived interactivity, relationship quality and psychological ownership. *Technological Forecasting and Social Change*, 175(November 2021), 121389. <https://doi.org/10.1016/j.techfore.2021.121389>
- TheR1D. (2023). *Shell GPT: Command-line productivity tool powered by ChatGPT*. GitHub. https://github.com/TheR1D/shell_gpt%0A%0A
- Uddin, M., Irshad, M. S., Kandhro, I. A., Alanazi, F., Ahmed, F., Maaz, M., Hussain, S., & Ullah, S. S. (2025). Generative AI revolution in cybersecurity: a comprehensive review of threat intelligence and operations. In *Artificial Intelligence Review* (Vol. 58, Issue 8). <https://doi.org/10.1007/s10462-025-11219-5>

Wang, P., & Johnson, C. (2024). The impacts of generative artificial intelligence (AI) in knowledge discovery and generation for cyber defense. *Issues in Information Systems*, 25(2), 230–243. https://doi.org/10.48009/2_iis_2024_119

Zaydi, M., & Maleh, Y. (2025). GAI-Driven Offensive Cybersecurity: Transforming Pentesting for Proactive Defence. *International Conference on Information Systems Security and Privacy, I(Icissp)*, 426–433. <https://doi.org/10.5220/0013378700003899>