

Essay Spring 2025

PenTestGPT in Web Application Penetration testing

Master essay, spring 2025

Julia Vister
juliavi@ifi.uio.no

Department of Informatics
Faculty of Mathematics and Natural Sciences

29th June 2025



Julia Vister

Assessing the reliability of PenTestGPT

AI-Driven Web Application Testing

Supervisors:

Tamas György Bisztray, Rebeka Toth

Abstract

This thesis explores the field of artificial intelligence in penetration testing in web applications, focusing on the effectiveness of PenTestGPT, which is an AI-powered penetration testing tool. With the rise of AI-driven cybersecurity tools, the help of large language models in web application penetration testing remains an open question. Therefore, this thesis aims to conduct a case study in which PenTestGPT and human penetration testers independently assess vulnerable web applications. The findings aim to provide information on the usefulness of integrating AI tools into security testing and its potential role in automated web application security assessments.

Keywords: AI in penetration testing, automated testing, PenTestGPT, Web Application penetration testing, Reconnaissance

Contents

	Abstract	2
1	Introduction	7
	1.1 Motivation	8
	1.2 Objective	8
	1.3 Research Questions	8
	1.4 Contribution	8
2	Theoretical Background	9
	2.1 Large Language Models (LLM)	9
	2.2 Penetration Testing	11
	2.2.1 Definition and Purpose	11
	2.2.2 Types of Penetration Testing	11
	2.2.3 Phases of Penetration Testing	12
	2.2.4 Common Tools and Techniques	13
	2.2.5 Web Application Penetration Testing	14
	2.2.6 Legal and Ethical Considerations	16
	2.2.7 Threats to validity	17
	2.3 AI in cybersecurity	17
	2.3.1 AI-driven Pentesting: PentestGPT	17
3	Related work	19
	3.0.1 AI driven penetration testing	20
	3.0.2 Traditional and hybrid penetration testing	20
	3.0.3 Manual vs Automated testing: strengths and limitations	20
	3.0.4 Literature search strategy	21
4	Methodology	23
	4.1 Research Design	23
	4.2 Test Environment	23
	4.2.1 Web Application Design	23
	4.2.2 Vulnerability Structure	23
	4.3 Participant Groups	24
	4.3.1 Ethical Approval and Consent	24
	4.4 Tools and Procedures	24
	4.4.1 Task Execution	24
	4.4.2 Data Collection	24
	4.5 Planned evaluation metrics and acceptance criteria	24

Contents

List of Figures

2.1	Large Language Model Capabilities	9
2.2	Architecture of a large language model.	10
2.3	An overview of pen testing stages, with six more defined phases.	13
2.4	Screenshot of a Nmap CLI command and the output, showing open port, service, and version of a website used for an ethical hacking course at UiO. This command specifies to look for open ports in a specific range (port 4500-5000). The output tells us that port 4746/tcp is open, that it is a ftp service and the version is ProFTPD.	15
2.5	Passive reconnaissance from a CTF challenge using ViewDNS.info to examine the IP history of bmw.com. This reveals previous IP addresses, their geographical locations, and hosting providers which is useful for infrastructure mapping during early pentesting stages.	15
2.6	The PentestGPT modules and how they correlate to each other	18
3.1	PRISMA Flow Diagram of Study Selection Process. This diagram illustrates the systematic review process: records identified from database searches, screened, and included or excluded based on PRISMA 2020 guidelines. . .	21

List of Figures

Chapter 1

Introduction

In an era where digital infrastructure supports nearly every aspect of modern society from user interface to cybersecurity, web applications represent both a basis for technological progress and a significant vector for cyber threats. As organizations increasingly rely on complex web-based systems to handle sensitive data and provide critical services, securing these applications has become a top priority. In particular, AI-driven tools are increasingly being developed and supported in the field of penetration testing and vulnerability assessment, promising to automate complex tasks that originally require deep human expertise. As cyberattacks grow, not only in frequency but also in sophistication and automation, there is also a growing pressure to secure an organization's systems. This is further amplified by regulations, compliance requirements, and increased social awareness of digital privacy and risk. Penetration testing remains one of the most effective strategies in cybersecurity practice. It enables organizations to identify and remediate vulnerabilities before they are exploited by malicious actors by simulating real-world attacks. However, traditional penetration testing is resource intensive and requires skilled professionals to navigate systems, identify vulnerabilities, analyze responses, and think creatively about how to bypass protections. Among its many phases, the reconnaissance phase is particularly critical. It sets the foundation for all subsequent actions by gathering information about the target structure, technologies, and attack surfaces. Wrongdoings or omissions during this phase can result in critical vulnerabilities being missed altogether. Recently, tools like PentestGPT have advanced, which is built on large language models (LLMs) and is designed to imitate the reasoning process of penetration testers. Such tools hold great promise for automating reconnaissance tasks such as OSINT and service enumeration. However, despite growing attention, we still lack systematic evidence on the actual performance and limitations of these AI tools in realistic settings. Can an LLM-powered system like PentestGPT effectively replicate subtle decision-making and situational awareness of a skilled human tester? How does it perform when faced with web applications of varying complexity? In which scenarios can AI tools support, or even replace, human testers?

This thesis investigates these question by comparing the performance of PentestGPT against human penetration testers during the reconnaissance phase of web application penetration testing. A controlled experimental setup is developed consisting of web applications with varying complexity and vulnerability profiles. These applications are tested by both human participants, who range from certified professionals to ethical hacking students, and PentestGPT. The goal is to compare performance across different recon tasks, identify where AI can effectively support or replace human testers, and

understand the limits of current AI-driven approaches.

1.1 Motivation

The use of AI in cybersecurity, especially for offensive tasks such as penetration testing, is gaining significant traction in both industry and research. This is driven by a desire to increase efficiency, reduce costs, and scale testing efforts in environments where human testers may be limited. AI promises not only speed, but also the potential to identify overlooked patterns or vulnerabilities that static tools or tired eyes might miss. However, with this promise comes a critical need for validation. As LLMs become more capable, there is increasing interest in deploying them for security measures, though it remains unclear how well they perform in the real world. Although AI shows potential in isolated or lab-controlled tasks, we lack comparative evidence of effectiveness and limitations when used in actual web application security testing scenarios.

This thesis aims to address that gap by offering a practical and empirical evaluation of PentestGPT in a controlled, realistic setting and directly comparing it with human performance during the crucial reconnaissance phase. In doing so, it contributes to a more informed understanding of where AI stands today in relation to human expertise in penetration testing.

1.2 Objective

The goal of this study is to evaluate the extent to which the reconnaissance phase of web application penetration testing can be automated or assisted using a large language model - specifically, PentestGPT - compared to human testers of varying experience levels.

1.3 Research Questions

- RQ1: To what extent can large language models like PentestGPT effectively replicate the reconnaissance tasks performed by human penetration testers, and how does their performance compare across varying levels of human expertise?
- RQ2: In which specific reconnaissance tasks does PentestGPT show strengths or weaknesses compared to novice and expert human penetration testers?

1.4 Contribution

By exploring the intersection of manual expertise and intelligent automation, this research aims to contribute both practical insights for security practitioners and theoretical grounding for future studies in AI-driven penetration testing.

Chapter 2

Theoretical Background

This chapter will provide the theoretical background that is needed to understand this thesis. The chapter starts with an overview of large language models in Section 2.1, followed by a discussion of what penetration testing is in Section 2.2. AI in cybersecurity will be covered in Section 2.3. Lastly, related works will be explored in Section 2.4.

2.1 Large Language Models (LLM)

Large Language Models belong to the category of foundation models, which are neural networks that are trained on a massive set of data to supply the fundamental components necessary to enable multiple use cases and application scenarios. Because of this, LLMs are capable of understanding and generating human-like language, as well as other forms of content to handle diverse tasks. LLMs are able to understand context, produce coherent and contextually relevant content, translate languages, summarize text, answer questions, help writing, and also assist in code generation. Some well-known and easily accessible LLMs examples are Open AI's Chat GPT models or Google's BERT model. [9] Not only are LLMs helpful to a single individual with day-to-day tasks, they are also revolutionizing applications in a multitude of fields such as chat bots, research assistance, and content creation.

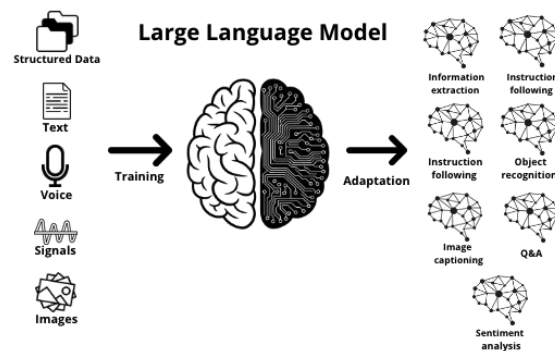


Figure 2.1: Large Language Model Capabilities

Large language models use deep learning techniques and huge amounts of textual data to function. They are built from multiple layers of neural networks each containing a parameter which can be fine-tuned. An attention mechanism further enhances these layers by making the model focus on relevant parts of the input. During the training, the model learns to predict the next word in a sentence by analyzing context and assigning probability scores to the likelihood of words - tokens - appearing. These tokens are then transformed into embeddings, numeric representation of the context. Once trained on training data, it can generate text by autonomously predicting the next word based on the input it receives. [9]

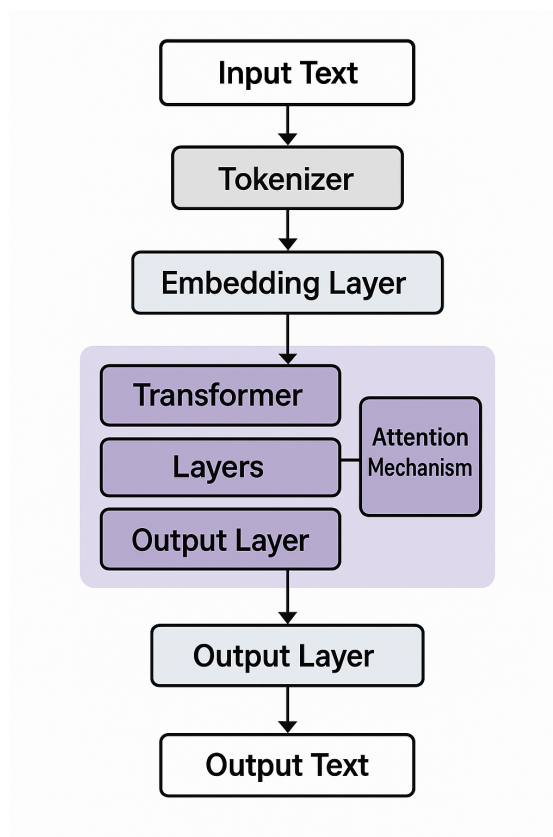


Figure 2.2: Architecture of a large language model

2.2 Penetration Testing

Penetration testing, often referred to as pentesting, is a technique used to identify security vulnerabilities, primarily within organizations. A penetration test simulates a cyberattack to find vulnerabilities in a computer system. By staging such an attack, a company's cybersecurity team can uncover critical security vulnerabilities and address them. This task is performed manually by experienced specialists known as penetration testers. They are professionals in ethical hacking, which is using hacking tools and techniques to fix and strengthen security weaknesses rather than causing harm. The phrase "penetration testing" describes one of the techniques utilized by ethical hackers. [10] Penetration testing should not be mistaken for vulnerability assessment, as the two serve distinct but similar purposes. While a vulnerability assessment is typically automated, performed regularly, and designed to detect common weaknesses, a penetration test goes beyond this routine scan. It simulates real-world attacks to uncover deeper, more complex security issues.

2.2.1 Definition and Purpose

There are three primary motivations for performing a penetration test.

In penetration testing, once vulnerabilities are identified, testers exploit them in simulated real-world attacks that mimic the behaviors of malicious hackers. This results in a more detailed and in-depth understanding of how attackers actually exploit vulnerabilities to gain unauthorized access to sensitive data or disrupt operations. With this in mind, an organization's security team can use the mock attack to design effective network security controls and defenses tailored to counter actual cyberthreats.

Pentesting combines automated tools and manual techniques to discover known and unknown vulnerabilities. In this case, pen testers actively attempt to exploit the weaknesses they find and, therefore, minimize false positives. If a pen tester can successfully exploit a vulnerability, so can cybercriminals. In addition, pen testing aids in meeting regulatory compliance by demonstrating that an organization's security controls function correctly. This ensures compliance with standards such as the General Data Protection Regulation (GDPR) and ISO standards.[1]

2.2.2 Types of Penetration Testing

There exist different types of pen tests that target different types of assets. The first one is the most crucial to understand for this thesis.

1. Application pen tests
2. Network pen tests
3. Hardware pen tests
4. Personnel pen tests

Application pen testing looks for vulnerabilities in applications and related systems; this also includes web applications, websites, application programming interfaces (APIs), mobile apps, and cloud apps. The most common method to use in this case is to look for known vulnerabilities listed in the OWASP Top 10, which is a list of the most critical vulnerabilities in web applications. [16] [10] In addition to this, it is common to look for less known security flaws that are unique to the app at hand. When it comes to Web applications, pen testing targets open source and custom web applications to identify and exploit vulnerabilities in authorization, security configuration, and data protection mechanisms. Some exploit techniques used for this are SQL injection, cross-site scripting (XSS), or cross-site request forgery (CSRF). [24] It examines the infrastructure, design, and configurations of a web application. [2] As mentioned in subsection 2.2.1 regarding compliance with security standards, Web application pen testing helps comply with standards and regulations such as HIPAA, GDPR, PCI-DSS and SOC-2. [24]

Network pen testing is divided into internal and external tests, and mimics attacks on a company's computer network both physically and more remotely. Hardware pen testing revolves around finding vulnerabilities in devices that are connected to the network. Examples of this are laptops, mobile devices, and operational technology (OT). Here, a pen tester may look for software flaws and physical vulnerabilities. Personnel pen testing looks for flaws in employee's security routines. It is meant to check how vulnerable a company is to social engineering attacks, often using methods such as phishing. [10]

2.2.3 Phases of Penetration Testing

This process is quite extensive and complex and involves multiple stages. There is no standard answer for how long the process should take, as it all depends on various variables such as the objectives, approach, and complexity of the attack surface. It includes gathering information about the target system, also called reconnaissance, identifying potential entry points, exploiting vulnerabilities, and documenting the findings in a detailed report. [18] There exists different ways to categorize penetration testing phases, however, the most common is to divide it into four larger processes or six more defined processes. [2] First is a planning phase where a scope for the test is defined. This scope outlines what systems shall be tested, when it will happen, and the methods to use. Another thing the scope defines is the amount of information that pen testers will have available in advance. [10].

1. **Reconnaissance:** The phase most crucial to this research. Gathers information on target system using various hacking techniques. If the target is an application, part of the recon may be to study the source code. Another method here is to use open source intelligence (OSINT), which is reading public information. This step is crucial for gathering critical information for the exploitation phase. [2]
2. **Target Discovery:** Pen testers use the information collected during the recon step to identify exploitable vulnerabilities. Examples of this is either using tools, such as a port scanner, or developing a fake story for a social engineering pen test. [10]
3. **Exploitation:** The actual attack occurs. Various types of attacks may be tried depending on the target, vulnerabilities, and scope, and they target each

vulnerability discovered during the reconnaissance. Once a vulnerability is exploited, pen testers can dig deeper and access more of the vulnerability. This part is called "vulnerability chaining". [2] [10]

4. **Cleanup and reporting:** Pen testers clean up any traces that are left behind, and any changes to the infrastructure will be restored to the state before testing began. A detailed report on the mock attack will also be issued. The report should include a list of all vulnerabilities and exploits that are categorized according to a risk level, as well as recommendations for improvements. [2]



Figure 2.3: An overview of pen testing stages, with six more defined phases.

2.2.4 Common Tools and Techniques

The most common tools to perform a reconnaissance, discover vulnerabilities, and automate parts of the pen test process include [10]:

1. Specialized operating systems. The most popular is Kali Linux.
2. Credential-cracking tools, such as Hydra and John the Ripper.
3. Port Scanners, such as Nmap.
4. Vulnerability scanners and web vulnerability scanners, such as Burp Suite.
5. Packet analyzers/packet sniffers, such as Wireshark and tcpdump.
6. Penetration testing framework, Metasploit.

2.2.5 Web Application Penetration Testing

The web application penetration process builds upon general penetration testing methodologies, but is tailored to the intricacies of web-based systems. It generally follows the core phases outlined in subsection 2.2.3: planning, reconnaissance, discovery, exploitation and reporting. The planning phase remains the same with defining a scope, deciding which application pages need to be tested, and whether to perform internal, external, or both testing. This section emphasizes the reconnaissance phase, focusing on its steps, tools, and practical examples relevant to web applications.

Reconnaissance Phase

The reconnaissance phase is critical to understand the structure, components and weak points of a web application. Reconnaissance can be divided into passive and active reconnaissance. *Active reconnaissance* is when a pen tester directly interacts with the target to get information. While not commonly employed in legal penetration testing engagements, techniques such as watering hole attacks - where a malicious actor compromises websites frequented by the target through loopholes - represent advanced social engineering methods that fall under broader reconnaissance strategies. However, there are many other techniques used for active reconnaissance such as port scanning, network mapping, DNS enumeration and vulnerability scanning. *Passive reconnaissance* is the opposite of active, meaning pen testers collect information without interacting directly with the target. A pen tester collects public information about the target in this part, without alerting the target. Examples of passive recon are monitoring social media or public databases. Some techniques used for passive reconnaissance is search engine dorking, public data aggregation, social media analysis and email harvesting. [22] Common techniques and tools used to gather information about the target:

- **Nmap (port scanning):** A network scanner used to discovers open ports, services running on those ports, and the operating system (OS) of the target system.

Case: A company focusing on tokenization services wanted to test two newly created web applications to find out if there are any security gaps. Nmap was used to scan the web application's infrastructure. This revealed open ports and services, including outdated versions of SSH and HTTP services. [11]

- **DNS enumeration:** Techniques and tools that uncover subdomains, domain structure, and misconfigured records.

Case: A multinational retail chain requested a security assessment on several web applications. Testers performed DNS enumeration to identify subdomains and uncovered legacy subdomains which pointed to deprecated services. [11]

- **Gobuster / Dirb:** Tools used to brute-forcing directories and file paths to discover hidden resources within web applications.

Case: In the same security assessment for the tokenization services provider, Dirb was used to brute-force directories on their web server. This led to the discovery of hidden administrative directories. [11]

- **SSL/TLS analysis:** Identifies weaknesses or misconfigurations in HTTPS setups, such as outdated protocols or missing certificate chains.

Case: An asset management company wanted a penetration test for their web application as they deal with a lot of sensitive data. It was revealed through

SSL/TSL analysis that the application was using deprecated TLS 1.1 protocols. [11]

- **WHOIS lookup:** Reveals domain registration and hosting information including contact details and infrastructure ownership. For example, a WHOIS lookup on a misconfigured server can reveal outdated administrative contacts and expired certificates - offering insights into organizational oversight and possible attack vectors.

Case: In a security evaluation for a US contract services company, WHOIS lookups - as part of open-source intelligence, were performed to gather information about domain registrations. This helped reveal contact information associated with the domain. [11]

- **Search Engine Dorking:** Utilizes advanced search operators (e.g., 'site:', 'intitle:', 'filetype:') to uncover exposed information indexed by search engines. Examples include login pages, configuration files, and backups.

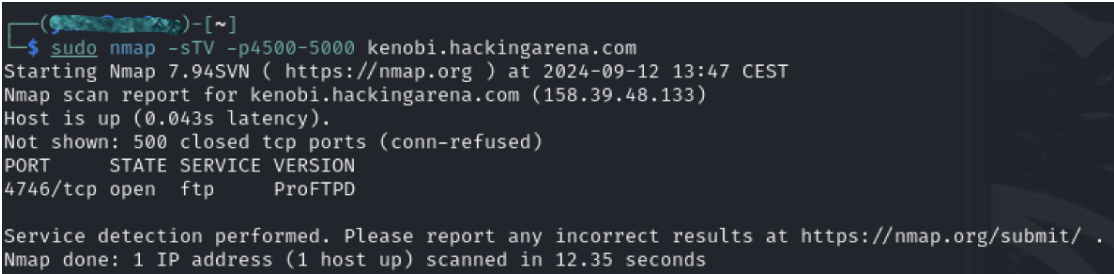


Figure 2.4: Screenshot of a Nmap CLI command and the output, showing open port, service, and version of a website used for an ethical hacking course at UiO. This command specifies to look for open ports in a specific range (port 4500-5000). The output tells us that port 4746/tcp is open, that it is a ftp service and the version is ProFTPD.



Figure 2.5: Passive reconnaissance from a CTF challenge using ViewDNS.info to examine the IP history of bmw.com. This reveals previous IP addresses, their geographical locations, and hosting providers which is useful for infrastructure mapping during early pentesting stages.

Vulnerability Scanning

Automated tools are used to scan web applications for known vulnerabilities. These tools cross-reference the application's code, configuration, and dependencies against Common

Vulnerabilities and Exposures (CVEs) databases to identify potential weaknesses. This helps identify security misconfigurations and outdated components. This part is called vulnerability scanning. [24] Common vulnerability scanners are: [23]

- **Nikto:** An open-source scanners that can identify vulnerabilities specific to web applications, such as Cross-Site Scripting (XSS) and SQL injection.
- **Burp Suite:** security testing platform and proxy tool that allows pen testers to intercept and modify web traffic. Facilitates brute-forcing and fuzzing attacks. [14]

Exploitation and Post-exploitation

Once the vulnerability scan is completed, manual exploitation is carried out to identify more complex or hidden vulnerabilities. This involves using specialized tools and techniques to gain unauthorized access to the system. Once a vulnerability is successfully exploited, pen testers can attempt to chain exploits together to achieve greater impact. Specific tools are often used to help exploit particular types of vulnerabilities, such as: [24]

- **SQLmap:** An automatic SQL injection and database takeover tool which can identify and exploit SQL injection vulnerabilities.[14]
- **Metasploit:** A widely used exploit framework with a library of exploits and payloads for known vulnerabilities which allows controlled compromise of systems.
- **Hydra:** A tool for performing brute-force attacks on login forms and protocols like HTTP, FTP and SSH.

2.2.6 Legal and Ethical Considerations

It is important to comply with and understand the legal requirements and restrictions that monitor penetration testing, which vary across regions. One of the most important legal considerations is the compliance with data protection laws, such as the GDPR in the European Union. This requires obtaining consent, anonymizing data, and ensuring secure handling of any personal or sensitive information. Ethical penetration testing also respects intellectual property rights and follows professional guidelines to ensure proper authorization and confidentiality.[8] With the integration of AI-driven tools like PentestGPT, it is important to note additional concerns:

- **Model bias and hallucinations:** AI models can reflect training biases, generate misleading outputs and therefore produce flawed reconnaissance insight. [25]
- **Malicious re-purposing:** The technologies that are supposed to assist testers can also be used for malicious purposes and LLMs are being exploited via malicious versions.
- **Prompt injection and security risks:** LLMs are vulnerable to attacks like prompt injection, which leads to the manipulation of models into doing unintended actions. [6]
- **Transparency:** LLM reasoning is often opaque, and complicates the audibility and trustworthiness in security situations. [25]

2.2.7 Threats to validity

Table 2.1: Threats to Validity in PentestGPT Evaluation

Threat	Description	Mitigation Strategy
Internal validity	Self built test environments might lack real-world complexity.	Use diverse environments with varying difficulty, replicate across multiple setups.
External validity	Findings may not generalize to large-scale or enterprise systems.	State limitations and encourage validation on real-life systems.
Construct validity	Evaluation focuses on task completion, not overall exploit success	Frame results as partial success in recon phase, not full exploitation.
Tool and model limitations	Accuracy relies on LLM training data, context window, and is prone to prompt injection.	Use prompt hygiene, fixed context formatting, and document failure cases.
Research bias	Manual tester skill level may influence comparisons.	Include testers of varied experience.
Ethical bias	LLMs may propose ethically questionable steps without context.	Apply strict prompt guidelines

2.3 AI in cybersecurity

2.3.1 AI-driven Pentesting: PentestGPT

PentestGPT is an LLM-powered automated penetration testing framework that is open source on GitHub and created by a university student who names himself GreyDGL.

"It is built on top of ChatGPT API and operate in an interactive mode to guide penetration testers in both overall progress and specific operations" - GreyDGL, creator of PentestGPT [17]

PentestGPT is designed with three modules: parsing, reasoning, and generation.[4] The parsing module analyzes the output of the penetration testing tools and the content displayed on the Web interface and, therefore, acts as a support interface. This module handles four types of information. The first (1) is user intentions - instructions provided by the user to guide subsequent actions. The second (2) consist of raw outputs generated by the security testing tools. The third (3) is raw HTTP web data derived from HTTP web interfaces, and the fourth (4) includes source codes extracted during the penetration testing process. The test reasoning module interprets the test results to guide penetration testers on the next appropriate actions. It receives the test results and prepares a testing strategy for the next steps. This strategy is then passed on to the generation module. The test generation module produces precise penetration testing commands and procedures that users can execute. Serving as a bridge between the reasoning module and subsequent actions. The generation module translates sub tasks from the reasoning module into concrete commands and actionable steps. In addition, it generates human-readable outputs to clearly convey the testing process. [4]

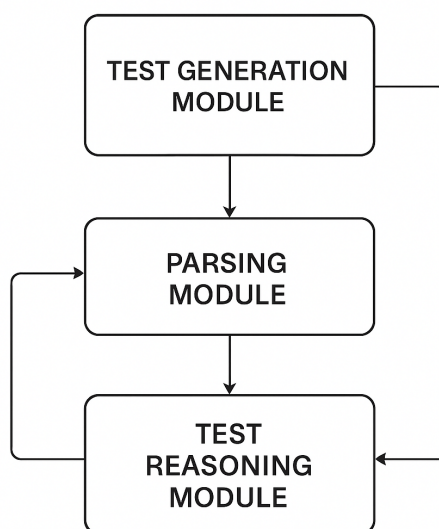


Figure 2.6: The PentestGPT modules and how they correlate to each other

Recent research demonstrates that PentestGPT effectively performs structured reconnaissance tasks within web application penetration testing. In comparative benchmarks using real-world vulnerable environments, the tools showed strong proficiency in specific sub-tasks such as invoking and interpreting outputs from tools like Nmap and Dirb, and also formulating appropriate next steps. [3] [19] [12]. For instance, the model excels at parsing Nmap scan results to identify open services, suggesting targeted enumeration commands, and analyzing directory brute-force tool outputs to map hidden files or subdomains. [5]

Additionally, PentestGPT is designed with a modular architecture—with separate reasoning, parsing, and generation components—that mitigates the common context-loss issues experienced by LLM-driven pentesting models. [13]. Further validation from the LLM-based penetration testing literature emphasizes similar capabilities. Isozaki et al.’s benchmark study with PentestGPT and open-source LLMs noted that enumeration tasks, which include structured activities like DNS record analysis, header inspection, and directory enumeration, represent an area where LLMs maintain a high success rate. [12]. Similarly, independent studies using ChatGPT (a general-purpose LLM) have shown that LLMs can reliably extract reconnaissance intelligence such as IP ranges, identified technologies, open ports, and SSL/TLS configurations. [21]

PentestGPT’s natural language capabilities allow it to translate technical findings into actionable intelligence, which is especially valuable for less experienced testers. However, PentestGPT’s usefulness may be limited in dynamic or ambiguous situations where creativity, intuition, or contextual judgment is required. [3]

Chapter 3

Related work

The reliance on artificial intelligence has increased greatly in cybersecurity, leading to advances in automated penetration testing. Traditionally, penetration testing is a time-consuming and costly process that involves human security experts who identify and exploit vulnerabilities manually. AI-driven approaches that use large language models such as PenTestGPT have emerged as alternatives to human-led security testing. These models claim to automate reconnaissance, vulnerability detection, and exploitation. However, this raises questions about their effectiveness, reliability, and limitations compared to human penetration testers. This review of the literature will look at existing research on AI-driven penetration testing, identify knowledge gaps, and establish a foundation for further research of LLM in security assessments.

3.0.1 AI driven penetration testing

Artificial intelligence (AI) and its technologies have the ability to increase the efficiency and effectiveness of identifying vulnerabilities and performing an assessment. Businesses adopting AI technologies for security testing using machine learning and known automated tools can simplify the assessment process, reduce human error, and provide more accurate information on their system's vulnerabilities.

Machine learning algorithms are the core of automated penetration testing. This is because these algorithms are able to combine and analyze large datasets to identify patterns and also predict potential vulnerabilities based on data given to the algorithm. This paves the way for AI tools to improve their detection algorithm over time, leading it to become effective in discerning attack vectors and new threats. Another critical piece in these AI tools is natural language processing, as this is what allows computers to read and analyze human language. [20]

3.0.2 Traditional and hybrid penetration testing

Although AI-driven tools offer promising improvements in penetration testing, traditional approaches remain widely adopted in practice, especially in security-sensitive sectors. An example is the finance sector where the transaction of money happens every day and hour. A recent study [7] proposed a framework that mimics financial institution websites to assess vulnerabilities using both manual and automated testing techniques. The researches implemented SQL injection attacks, cross-site scripting, and header analysis to identify faults. Tools such as OWASP ZAP and Nessus were used for automated validation. In particular, they observed minimal difference in results between manual and automated tests, supporting the idea that hybrid approaches can offer comprehensive coverage. This study also emphasized the cyclic nature of effective penetration testing, highlighting how vulnerability discovery was fed back into planning and mitigation efforts. Their proposed framework acts as a blueprint for financial web applications to identify and mitigate threats early in the development life-cycle. Although the focus was not on the reconnaissance phase alone, the information gathering component, including the use of Google dorks and SQL injection vectors, directly parallels early-stage penetration tasks. These traditional techniques serve as a baseline for evaluating newer, AI-powered approaches such as those performed by models like PentestGPT. [7]

3.0.3 Manual vs Automated testing: strengths and limitations

Another relevant contribution comes from Nagpure and Kurke [15], who offer a comparative overview of Vulnerability Assessment and Penetration Testing (VAPT), specifically emphasizing the limitations of automation. As web application attacks grow in frequency and sophistication, they argue for a hybrid security approach that balances automated tools with manual expertise. Their study highlights that while automated tools like Burpe Suite and OWASP ZAP efficiently detect common vulnerabilities - such as SQL injection, cross-site scripting, and directory traversal- they often miss more complex vulnerabilities such as authentication bypass, file upload flaws and Cross Site Request Forgery (CSRF). Their findings suggest that certain vulnerabilities such as CSRF may pass automated checks only if a superficial presence of protections -

like CSRF tokens - is detected. However, verifying whether these protections are correctly implemented often requires manual inspection and logical reasoning, areas where automated tools fall short. This distinction is vital to understanding the current capabilities, limits and potential role of AI-driven models like PentestGPT in web application reconnaissance tasks.

Reconnaissance tasks such as fingerprinting technologies, identifying exposed directories, or evaluating error messages can often be automated and are well-suited for this. Pattern recognition, deeper interpretations like detecting subtle logic flaws or poorly implemented security controls, require human intelligence and contextual reasoning. [15]

By evaluating both the strengths of automation in pattern recognition and the necessity of human insight in nuanced testing scenarios, this thesis positions PentestGPT within a realistic framework. Specifically, it investigates whether LLMs can perform reconnaissance at a level comparable to novice or expert testers, and which reconnaissance tasks are best suited to AI versus humans.

3.0.4 Literature search strategy

To ensure a structured review of relevant literature, a systematic search was conducted using Google Scholar and IEEE Xplore, focusing on works published between 2019 and 2024. Keywords included “AI in penetration testing,” “PentestGPT”, “automated testing” and related terms. After removing duplicates and screening based on relevance, X sources were selected for detailed review. The PRISMA-style flow diagram in Figure 3.1 outlines the selection process.

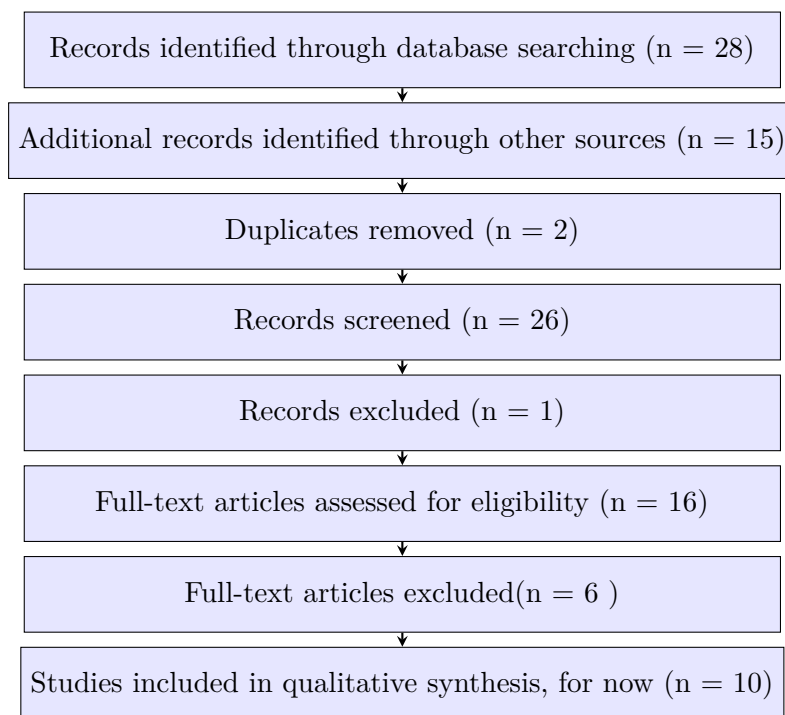


Figure 3.1: PRISMA Flow Diagram of Study Selection Process. This diagram illustrates the systematic review process: records identified from database searches, screened, and included or excluded based on PRISMA 2020 guidelines.

Chapter 4

Methodology

The goal of this chapter is to familiarize ourselves with the research methods used to answer the research questions presented in the introduction.

4.1 Research Design

This is a comparative experimental study that uses a quasi-controlled environment. Two groups - PentestGPT and human testers - will perform reconnaissance on prebuilt web applications with known vulnerabilities. Their findings will be compared quantitatively and qualitatively.

4.2 Test Environment

4.2.1 Web Application Design

Ten identical applications in layout and tech stack. Each version increases in complexity and obfuscation of vulnerabilities. There is varying security controls and detection difficulty. All applications will be internally documented with a ground truth list of vulnerabilities.

4.2.2 Vulnerability Structure

- Web application 1: Trivial vulnerabilities
- Web application 5: Intermediate complexity
- Web application 10: High complexity

Group	Description
A	Certified professional penetration testers
B	Hobbyists/CTF participants
C	Ethical hacking students at UiO

4.3 Participant Groups

Each participant will be assigned the same applications and allowed the same tools and time limits as PentestGPT.

4.3.1 Ethical Approval and Consent

Participants will be recruited voluntarily, anonymized and approved under university research ethics.

4.4 Tools and Procedures

4.4.1 Task Execution

All participants, including PentestGPT, are given a standardized instruction sheet. Fixed time budget per application, e.g., 30-60 minutes per application.

- Example task: "Perform reconnaissance on this web application and enumerate as much information as possible that could aid later exploitation".

4.4.2 Data Collection

- Human Participants: Record command history, notes, outputs, tools used, results.
- PentestGPT: Prompt logs and output.

Screen recording or activity logs to capture user sessions for review.

A

4.5 Planned evaluation metrics and acceptance criteria

To ensure a systematic comparison between PentestGPT and human testers, a set of planned evaluation metrics has been defined. These metrics are designed to assess the performance in terms of accuracy, completeness and tool usage. Acceptance criteria are established to help benchmark success and enable reproducibility of evaluation across all participant groups.

Table 4.1 summarizes the planned metrics and corresponding acceptance thresholds used to evaluate reconnaissance performance.

4.5. Planned evaluation metrics and acceptance criteria

Table 4.1: Evaluation Metrics and Acceptance Criteria

Metric	Description	Acceptance Criteria
Vulnerability Discovery Rate	Number of correctly identified vulnerabilities during reconnaissance	Compared against ground truth per application. Success if $\geq 70\%$ of listed vulnerabilities are discovered.
Information Enumeration Quality	Accuracy and depth of discovered assets, URLs, technologies, and endpoints	Minimum 70% match with ground truth. Precision and recall will be measured.
Tool Usage Effectiveness	Appropriate use of tools (e.g., Nmap, Dirb, WHOIS) and correct interpretation of outputs	At least 2 different tools used correctly; outputs interpreted without major errors
Time Efficiency	Time taken to complete the reconnaissance phase	Completion within 30–60 minutes depending on app complexity
Reproducibility of Output	How consistent results are when repeated or re-prompted (for GPT) or done by peers (humans)	$\geq 80\%$ of key findings reproduced in follow-up sessions
Identification of Critical Misconfigurations	Discovery of obvious misconfigurations or insecure defaults	At least one critical issue found in apps known to contain them

Bibliography

- [1] BlackDuck. *What is Penetration Testing and How Does It Work? | Black Duck*. Penetration Testing. 2024. URL: <https://www.blackduck.com/glossary/what-is-penetration-testing.html> (visited on 29/04/2025).
- [2] CyberCX. *The Complete Guide To Penetration Testing*. CyberCX. 2025. URL: <https://cybercx.com.au/resource/penetration-testing-guide/> (visited on 01/05/2025).
- [3] Gelei Deng et al. *PentestGPT: An LLM-empowered Automatic Penetration Testing Tool*. 2nd June 2024. DOI: 10.48550/arXiv.2308.06782. arXiv: 2308.06782[cs]. URL: <http://arxiv.org/abs/2308.06782> (visited on 21/01/2025).
- [4] Gelei Deng et al. “Pentestgpt: An llm-empowered automatic penetration testing tool”. In: *arXiv preprint arXiv:2308.06782* (2023).
- [5] denizhalil. *MD2PDF TryHackMe Walkthrough: Learn Cybersecurity through Web Application Exploitation*. DenizHalil - Professional Cybersecurity Consulting and Penetration Testing. 20th Jan. 2025. URL: <https://denizhalil.com/2025/01/20/mdp2pdf-tryhackme-walkthrough/> (visited on 29/06/2025).
- [6] Mohamed Amine Ferrag et al. “Generative AI in cybersecurity: A comprehensive review of LLM applications and vulnerabilities”. In: *Internet of Things and Cyber-Physical Systems* 5 (2025), pp. 1–46. ISSN: 2667-3452. DOI: <https://doi.org/10.1016/j.iotcps.2025.01.001>. URL: <https://www.sciencedirect.com/science/article/pii/S2667345225000082>.
- [7] Arvind Goutam and Vijay Tiwari. “Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application”. In: *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. 2019 4th International Conference on Information Systems and Computer Networks (ISCON). Nov. 2019, pp. 601–605. DOI: 10.1109/ISCON47742.2019.9036175. URL: <https://ieeexplore.ieee.org/abstract/document/9036175> (visited on 11/06/2025).
- [8] Andreas Happe and Jürgen Cito. *On the Ethics of Using LLMs for Offensive Security*. 10th June 2025. DOI: 10.48550/arXiv.2506.08693. arXiv: 2506.08693[cs]. URL: <http://arxiv.org/abs/2506.08693> (visited on 29/06/2025).
- [9] IBM. *What Are Large Language Models (LLMs)? | IBM*. 2nd Nov. 2023. URL: <https://www.ibm.com/think/topics/large-language-models> (visited on 22/04/2025).
- [10] IBM Security. *What is Penetration Testing? | IBM*. 24th Jan. 2023. URL: <https://www.ibm.com/think/topics/penetration-testing> (visited on 29/04/2025).
- [11] *Information Technology (IT) Case Studies - ScienceSoft*. URL: <https://www.scnsoft.com/case-studies/security-testing> (visited on 28/05/2025).

- [12] Isamu Isozaki et al. *Towards Automated Penetration Testing: Introducing LLM Benchmark, Analysis, and Improvements*. 21st Feb. 2025. DOI: 10.48550/arXiv.2410.17141. arXiv: 2410.17141[cs]. URL: <http://arxiv.org/abs/2410.17141> (visited on 29/06/2025).
- [13] He Kong et al. *VulnBot: Autonomous Penetration Testing for A Multi-Agent Collaborative Framework*. 23rd Jan. 2025. DOI: 10.48550/arXiv.2501.13411. arXiv: 2501.13411[cs]. URL: <http://arxiv.org/abs/2501.13411> (visited on 29/06/2025).
- [14] MSBJ. *10 Tools For Web Application Penetration Testing*. Medium. 21st July 2023. URL: <https://medium.com/@msbj/10-tools-for-web-application-penetration-testing-2c7ab11c2e7> (visited on 03/05/2025).
- [15] Sangeeta Nagpure and Sonal Kurkure. "Vulnerability Assessment and Penetration Testing of Web Application". In: *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). Aug. 2017, pp. 1–6. DOI: 10.1109/ICCUBEA.2017.8463920. URL: <https://ieeexplore.ieee.org/abstract/document/8463920> (visited on 11/06/2025).
- [16] OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: <https://owasp.org/> (visited on 28/05/2025).
- [17] *PentestGPT/README.md at 6a767aae02b12d36717f647dc27f4e912d1a20a4 · GreyDGL/PentestGPT*. GitHub. URL: <https://github.com/GreyDGL/PentestGPT/blob/6a767aae02b12d36717f647dc27f4e912d1a20a4/README.md> (visited on 28/05/2025).
- [18] R. S. I. Security. *What is the Penetration Testing Execution Standard?* RSI Security. 21st Mar. 2024. URL: <https://blog.rsisecurity.com/what-is-the-penetration-testing-execution-standard/> (visited on 28/01/2025).
- [19] Xiangmin Shen et al. *PentestAgent: Incorporating LLM Agents to Automated Penetration Testing*. 29th May 2025. DOI: 10.48550/arXiv.2411.05185. arXiv: 2411.05185[cs]. URL: <http://arxiv.org/abs/2411.05185> (visited on 29/06/2025).
- [20] Mariam Soltanifar. "AI-Driven Penetration Testing: Automating Vulnerability Assessments". In: *Journal of Computing and Information Technology* 2.1 (25th May 2022). Number: 1. URL: <https://universe-publisher.com/index.php/jcit/article/view/31> (visited on 21/01/2025).
- [21] Sheetal Temara. *Maximizing Penetration Testing Success with Effective Reconnaissance Techniques using ChatGPT*. 2023. arXiv: 2307.06391 [cs.CR]. URL: <https://arxiv.org/abs/2307.06391>.
- [22] Muhammad Usman. *Active and Passive Reconnaissance Techniques*. Accessed: 05.05.2025. URL: <https://tryhackme.com/resources/blog/active-passive-reconnaissance-techniques>.
- [23] *Vulnerability Scanning Tools | OWASP Foundation*. URL: https://owasp.org/www-community/Vulnerability_Scanning_Tools (visited on 03/05/2025).
- [24] *What is Web App Penetration Testing? [How to Conduct It]*. Section: Security Audit. URL: <https://www.getastra.com/blog/security-audit/web-application-penetration-testing/> (visited on 03/05/2025).
- [25] Jie Zhang et al. *When LLMs Meet Cybersecurity: A Systematic Literature Review*. 4th Dec. 2024. DOI: 10.48550/arXiv.2405.03644. arXiv: 2405.03644[cs]. URL: <https://arxiv.org/abs/2405.03644> (visited on 21/01/2025).