Secure Linux Server Configuration and Performance Evaluation Using Command-Line Administration

# Introduction

The current data centres have become major power consumers in the world, with an estimated 1 percent of the world's electricity consumption, and the estimates have been pegged at 3-8 percent by 2030. The second thing that can help mitigate this impact is an efficient arrangement of operating systems, where the optimized systems can cut the level of consumption to 15-30 percent with better resource utilization. This requires system administrators to have excellent open system configuration, security hardening, as well as performance testing skills.

This test is based on the implementation and administration of a Linux operating system in a headless server. All the administration is done remotely with the use of Secure Shell (SSH), imposing command-line expertise and relating to the practical aspects of the working environment. The coursework is divided into seven steps, which start with system planning up to security auditing systems and evaluation of performance.

During the evaluation, the security best practices like SSH hardening, firewall settings, mandatory access control, intrusion prevention, and auto-updates are applied. Performance testing is done with representative workloads to test the system behavior, memory and CPU under stress. The result is a safe, effective and efficiently managed Linux server that was fully set up through remote administration using commands.

# Phase 1: System Planning and Distribution Selection

**System Architecture Overview**

The architecture of the system used in this assessment has two different virtual machines that will implement professional command-line administration practices. The former is a headless Linux server with unsere-desktop Ubuntu Server. The tasks of this server are to host applications and implement security controls, as well as to perform performance assessments.



Figure 1: Virtual Machines running on VirtualBox

Oracle VirtualBox is used to deploy both virtual machines. Host-only or in-house network setup is applied to allow direct communication between the workstation and the server with no external connection to networks. As shown in Figure 1, the virtual machines under VirtualBox are followed by a concurrent running system in a dual system architecture.

**Network Structure and Organisation**

A base private IP address is assigned to the server to be able to provide stable and dependable access remotely. Details of the network configurations (IP addresses and interface settings) are confirmed with the use of the command-line programs such as ip addr. The IP setup of the server, as obtained after inserting the figure in Figure 2, confirms that the network was set up properly. Connection between the workstation and the server is verified with the help of the ping command that has been provided in Figure 3, and it can be stated that the communication and the network are stable.

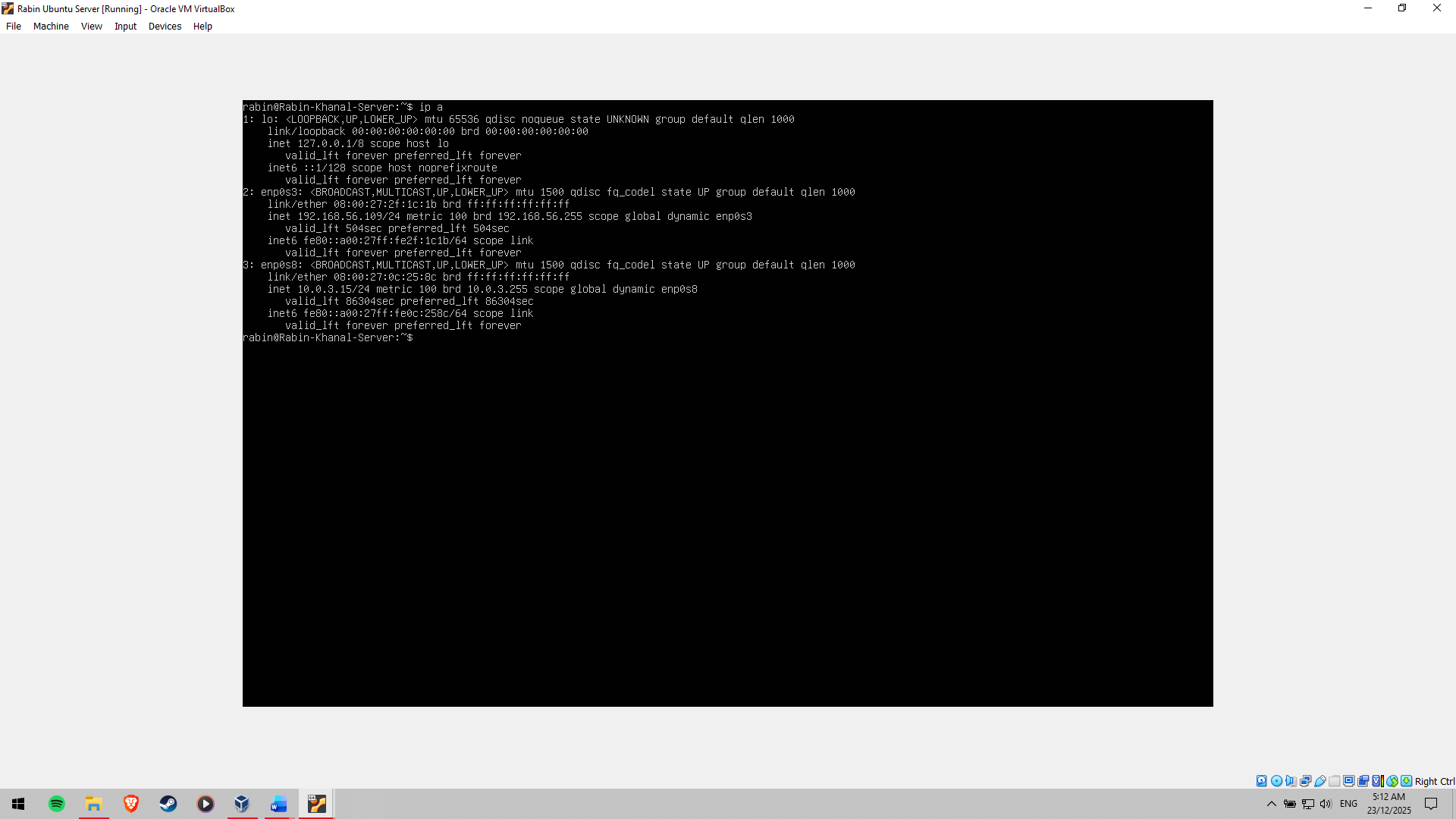


Figure 2: IP Configuration of the Server

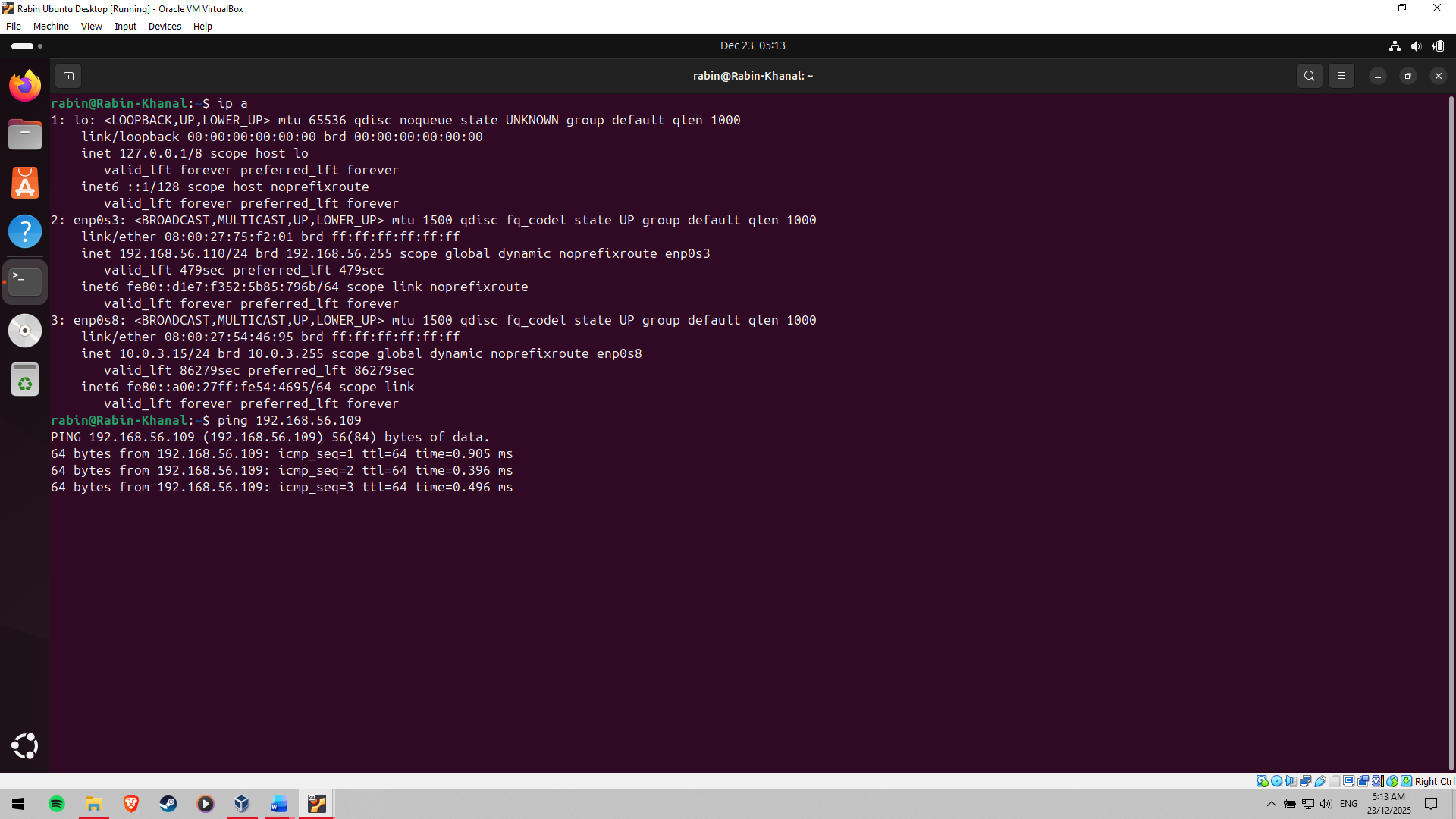


Figure 3: Ping from Desktop to Server successful

**Finding and choosing a Linux Distribution**

Ubuntu Server was chosen as the operating system for the server as it supports long-term support (LTS), is stable, and has a great deal of community documentation. Ubuntu Server offers a reasonable mix of security releases, bottleneck package management in APT and industry acceptance as compared to other competing systems like CentOS Stream and Debian. It supports AppArmor by default and unattended security upgrades, well in line with the security requirements in this assessment [1].

The workstation system has been set up to offer a secure SSH client system to be administrated remotely. This can be a Linux desktop virtual machine or the host operating system, at the will of the user. The selected configuration guarantees a stable SSH connection, the presence of monitoring systems, and a separation of administrative tasks and server infrastructure.

# Phase 2: Security Planning and Testing Methodology

**Methodology of Performance Testing**

A performance testing methodology is developed in a structured way and is aimed at testing the behaviour of the Linux server under different workloads, with the administrator being remotely located [2]. Every activity done in the monitoring is performed via the workstation system through SSH; this is to respond to the administrative limitations of the assessment. Monitoring tools include top, htop, vmstat, iostat and sar, which are command-line monitoring tools used to hand performance data to provide a snapshot of the machine. These monitors can analyse the CPU utilisation, memory usage, disk I/O activity and system load in real time and in history. Before deployment of applications, baseline measurements are taken to form a reference point to be used during testing of load and optimisation testing [3].

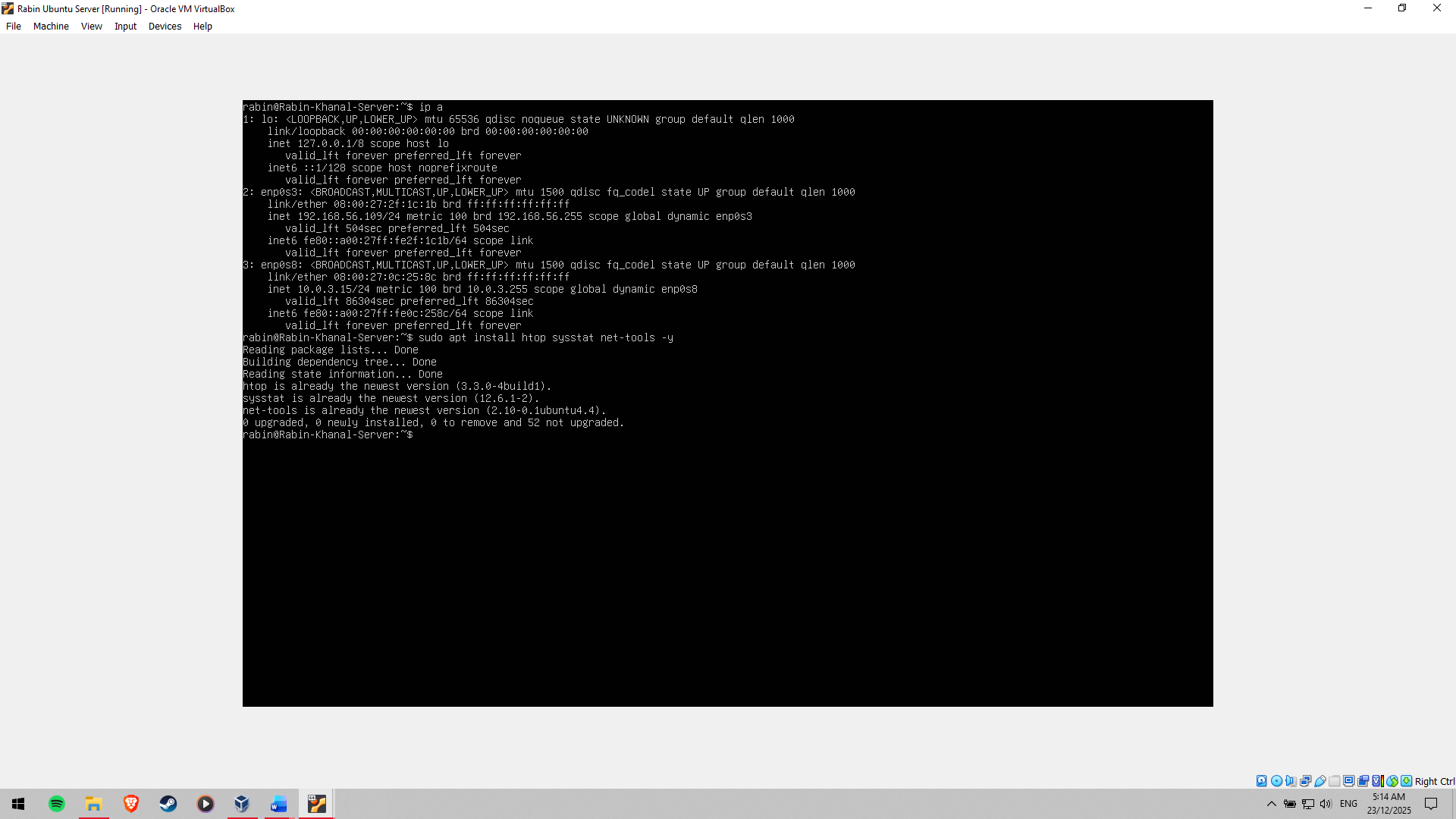


Figure 4: Tools for monitoring installed

**Security configuration checklist**

A comprehensive security baseline is created to reduce the attack surfaces and system integrity. Key-based authentication, disabling root log-in and blocking password-based access are the factors that put SSH hardening at the top priority. UFW is used to provide firewall protection, whereby only the authorised IP address of the workstation is allowed to access SSH [4]. AppArmor is used to impose Mandatory Access Control, which administers the application privileges as well as limits the potential damage that could be caused by compromised services [5]. The automatic security update option will be on to ensure that the vulnerabilities are patched in time. The management of user privileges is achieved by performing the creation of a non-root administrative user, which has regulated sudo access and minimises the risk posed with high privileges.

**Threat Modelling and Risk Mitigation**

A threat model is created to detect and reduce the major security threats. Brute-force SSH attacks can be considered one of the main threats, which can be challenged by providing key-based authentication, limited firewall filters, and installing fail2ban. The other threat is the escalating privilege within misconfigured user permissions, which is solved through the implementation of least-privilege principles and periodic auditing of sudo access. The third threat is the use of unpatched vulnerabilities, which are mitigated by automatic security updates and regular system audits. Collectively, these interventions create an expectation of a consistent security position and do not compromise system performance and usability [6].

# Phase 3: Application Selection for Performance Testing

**Application Selection Rationale**

Various applications with different consumption patterns under varying workloads are chosen to assess the performance of an operating system under different features. This strategy will make sure that CPU, memory, disk I/O, and network subsystems are assessed thoroughly. To compare system behaviour in both synthetic and service-oriented conditions and identify bottlenecks and optimisation opportunities, it is appropriate to choose different workloads and compare them effectively [7].

**Application Selection Matrix**

Stress testing tools like stress, stress-ng or similar software are used to represent CPU-intensive workloads and provide sustained load to the processor and observe scheduling efficiency or CPU utilisation. Memory-intensive workloads are modelled using tools that allocate and consume large amounts of system RAM, and allow management of memory management and swap behaviour to be evaluated [8]. To compute throughput and latency, disk I/O performance was called out using dd and fio, which create sequential and random read/write patterns. To test network-intensive workloads, iperf3 is used to test bandwidth, network probing, and network latency. Also, a lightweight server program, e.g. an HTTP server or a game server, is also deployed to simulate the workloads of actual service providers and test the responsiveness of the server under a load.

**Installation Documentation**

To meet administration restrictions, all applications are installed on the server by SSH-based command-line functions. The installation is done through the installation of the APT package manager, which provides a safe and verifiable deployment of the software. Installing commands is recorded precisely to facilitate reproducibility and auditing. As can be seen in Figure 5, all the applications chosen have been successfully installed on the server and can be tested.

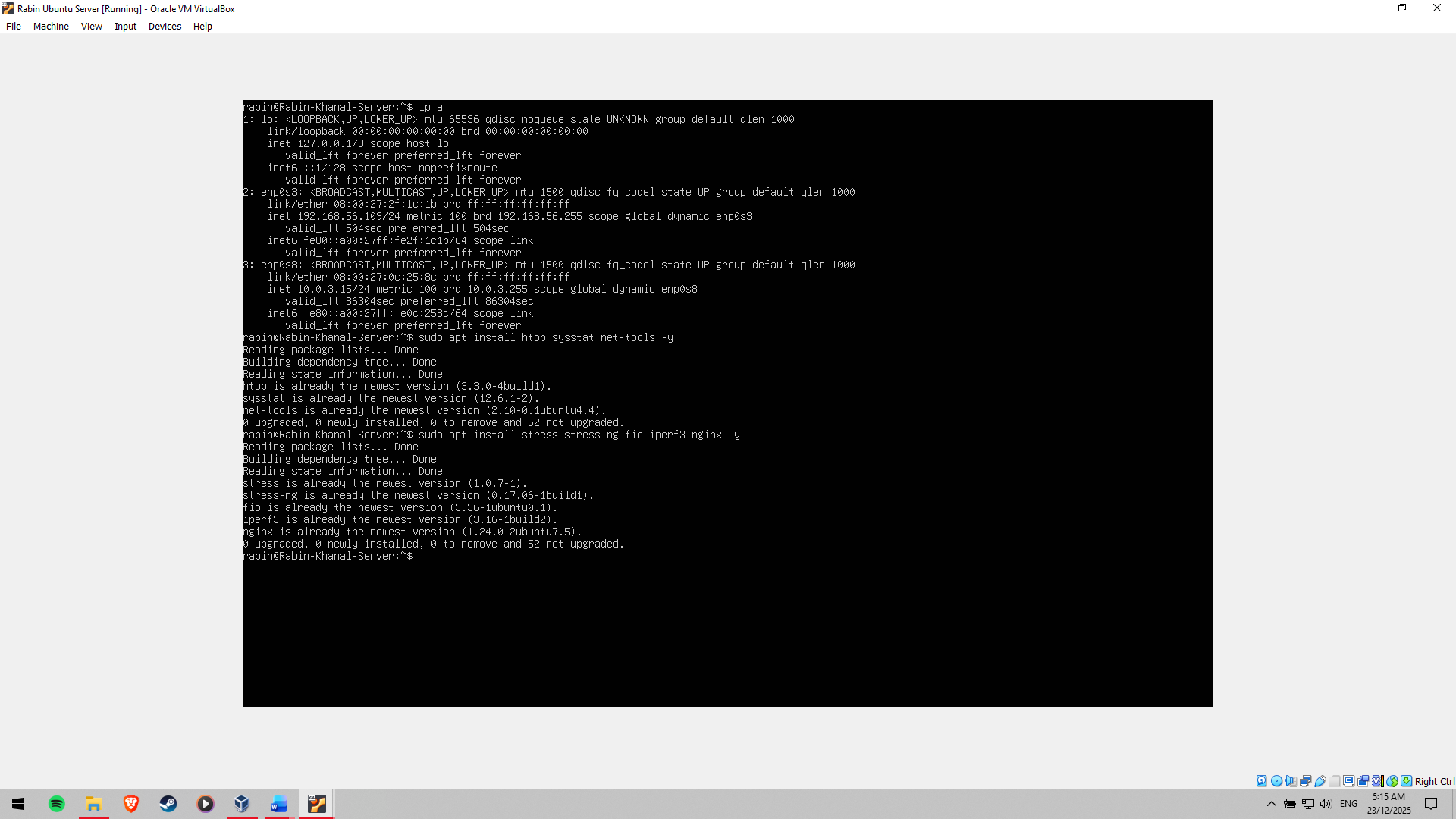


Figure 5: Applications for performance testing installed

**Expected Resource Profiles**

Every application cannot be used without a profile of the desired usage of the resources. CPU stress tools are expected to result in high processor active usage and little use of memory. The Intensive applications will consume a lot of RAM and probably lead to the use of swaps. The disk benchmarking tools will likely cause a persistent I/O load, which will influence the disk throughput and latency. The network testing tools should raise the network traffic and bandwidth. These are expectations upon which performance monitoring and analysis are guided.

**Monitoring Strategy**

Knowledge on monitoring performance of each application remotely is done using top, htop, iostat, vmstat, and iftop. The strategy guarantees that during testing, there is a consistent and accurate metering of the behaviour of the systems.

# Phase 4: Initial System Configuration & Security Implementation

**Installation and Secure Configuration**

The Linux server has Secure Shell (SSH) installed and enabled so as to be able to administer it remotely [9]. The service of SSH is also running, as displayed in Figure 6.

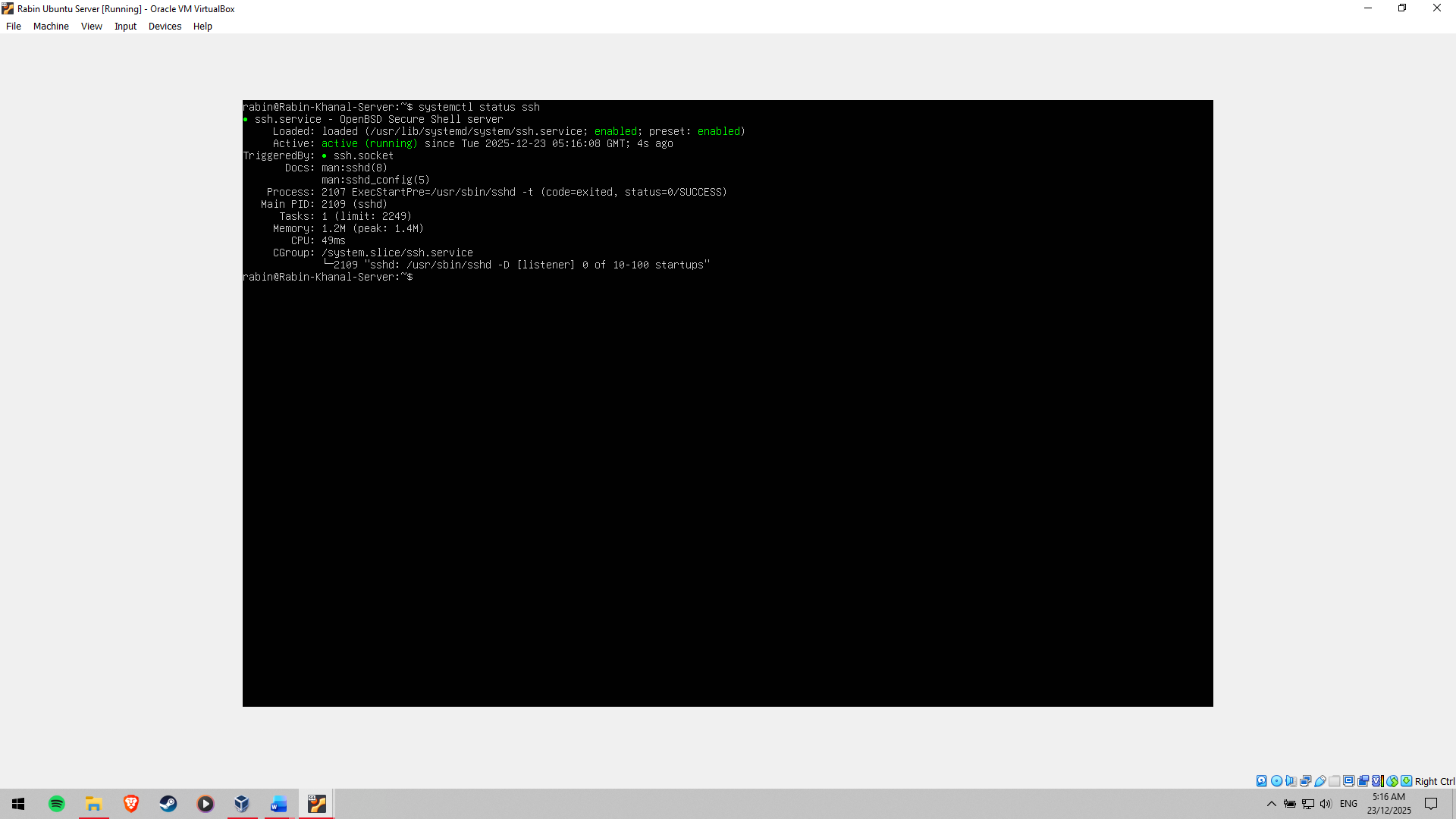


Figure 6: SSH installed and active on Server

To improve security, SSH is set to key-based authentication rather than password-based. A pair of SSH keys is created in the workstation, and the public one is safely passed into the server and placed in the authorised keys file as shown in Figure 7 and Figure 9. This would go a long way in avoiding the possibility of brute-force attacks.

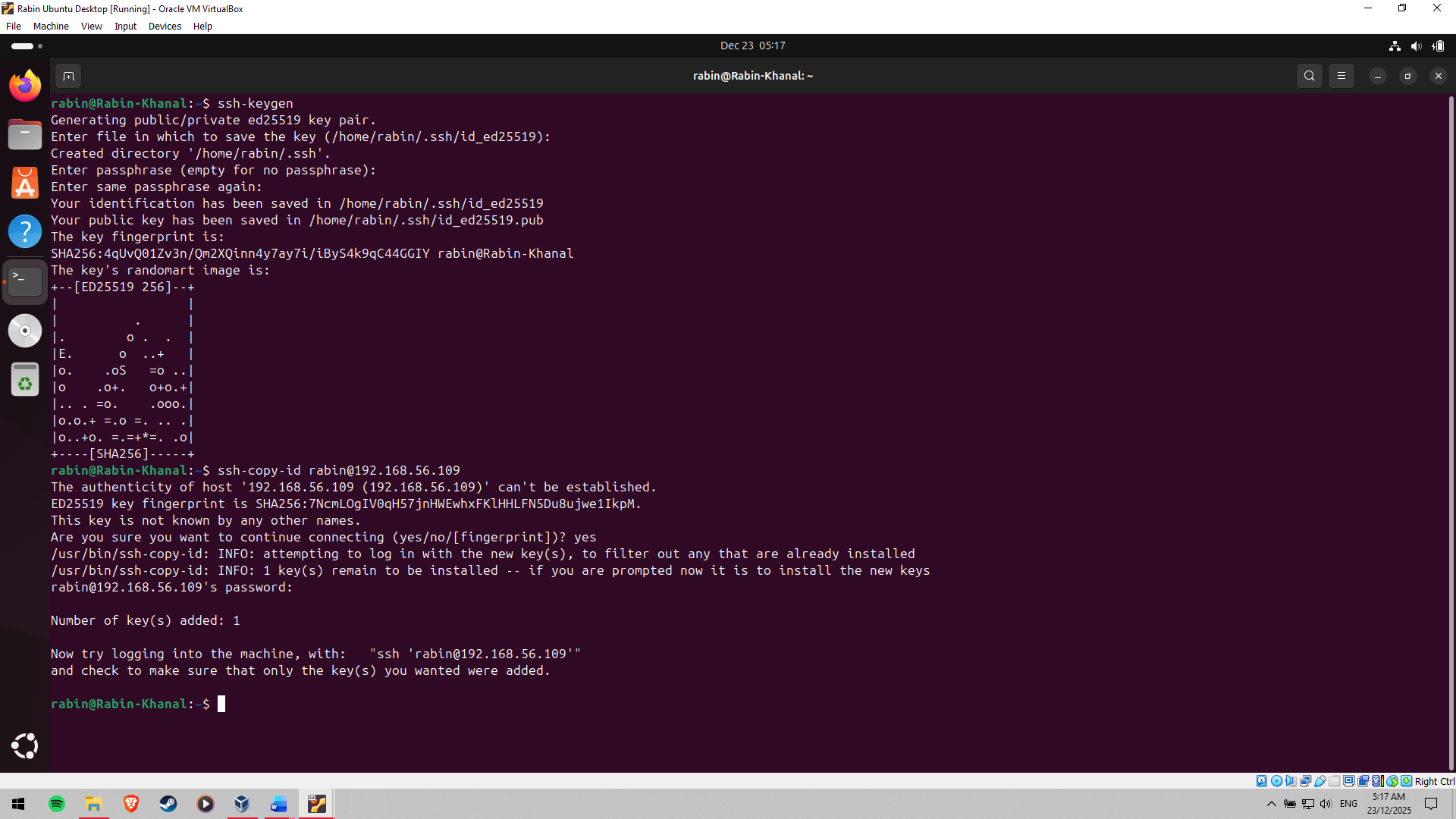


Figure 7: SSH key generated and public key added to the Server

**SSH Hardening Measures**

Additional SSH hardening is done by disabling root login and password-based authentication in the SSH configuration file. The changes also guarantee that access by administrators is available to authenticated users whose cryptographic keys are valid [10]. Figure 8 validates that root login and password authentication are switched off successfully, which imposes stringent policies of access controls.



Figure 8: Root Login and Password authentication disabled on Server

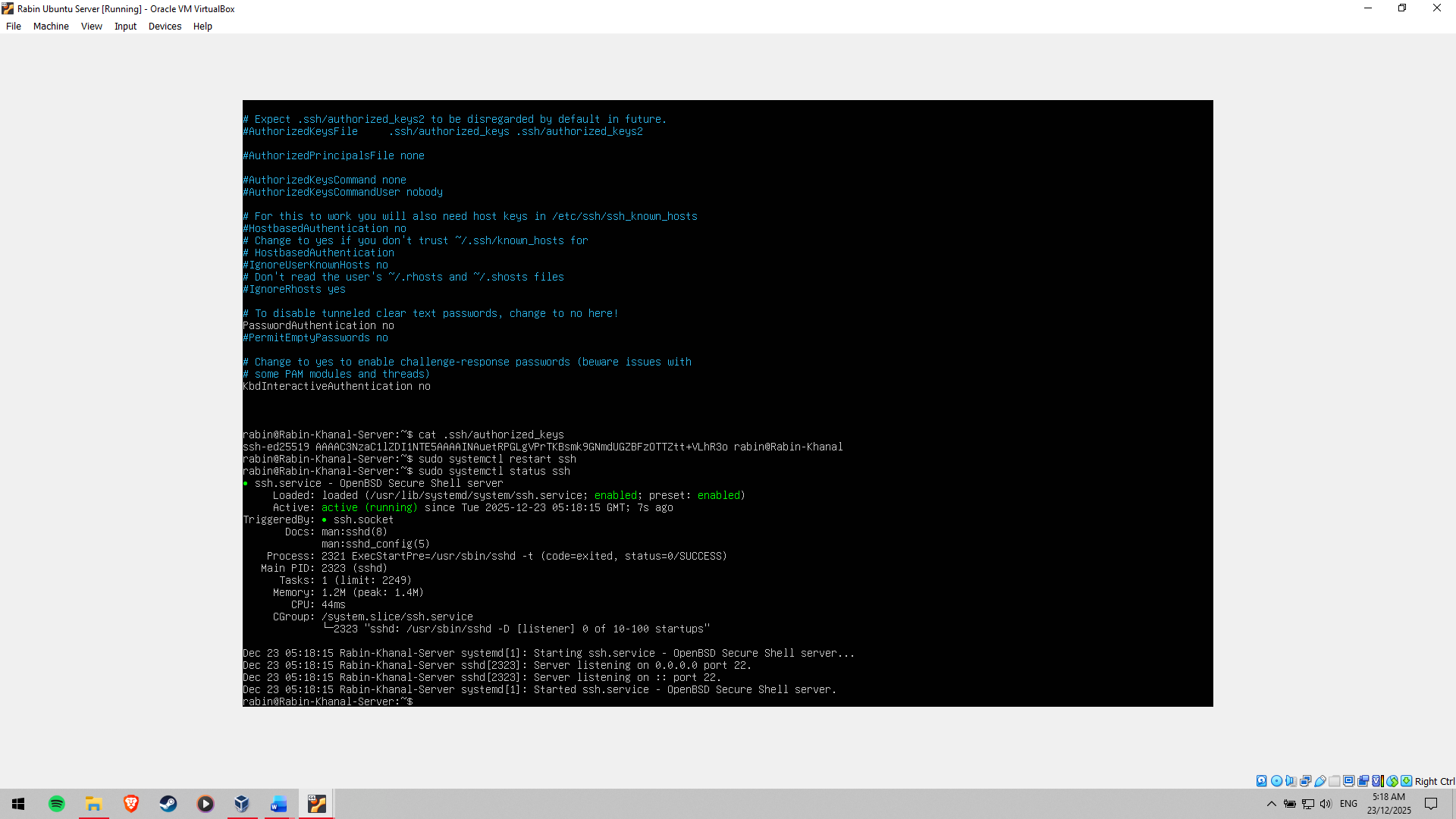


Figure 9: Authorised key present in the Server

**UFW Firewall set-up**

Uncomplicated Firewall (UFW) is used to configure a host -based firewall to regulate the traffic through the server. A default deny policy has been enabled in the firewall, allowing only necessary connections inbound. Only one workstation IP address that is authorised is allowed to access in a single SSH session, hence only the trusted sources are allowed to connect to the server. This is shown in Figure 10, which reveals that UFW is operational and that it has been properly set up to permit SSH connections under restrained conditions.

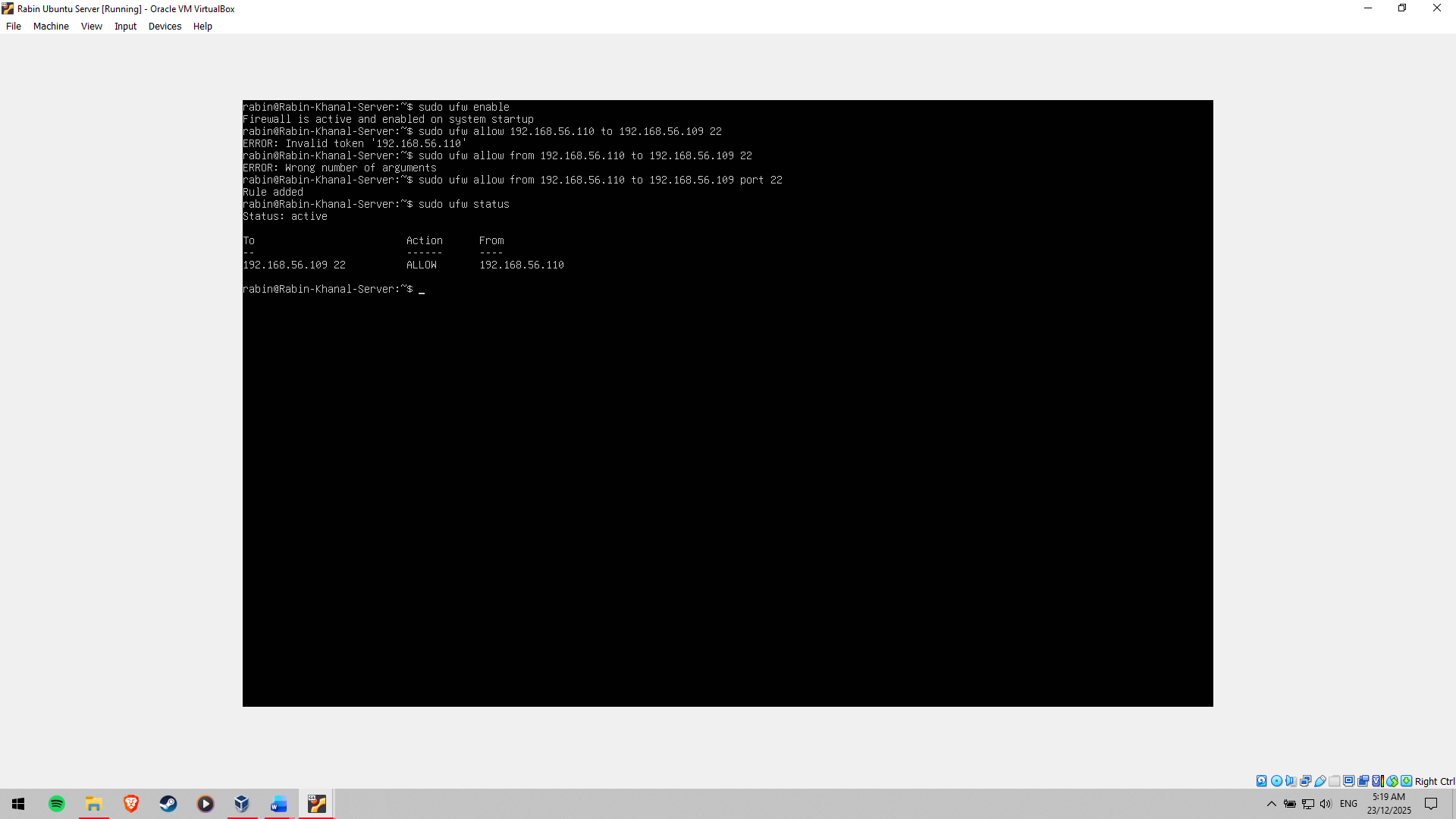


Figure 10: UFW active to allow SSH connection to the server

**User and Privilege Management**

In order to comply with the principle of least privilege, a non-root administrative user is launched to perform usual management tasks. This user will have unlimited sudo privileges, where he or she can take administrative actions without having to be in the root. This strategy minimises the risk of unintentional or malicious alterations of the system, and the ability to flex the system remains operational.

**Remote Administration Evidence**

The successful remote administration is confirmed by setting up the SSH connection between the workstation and the server, as in Figure 11. The entire system configuration is accomplished remotely through the use of SSH, which meets the requirements of the administrative constraints of the assessment. Changes in configuration files have a history of before and after comparison to show the transparency and traceability of security implementations.

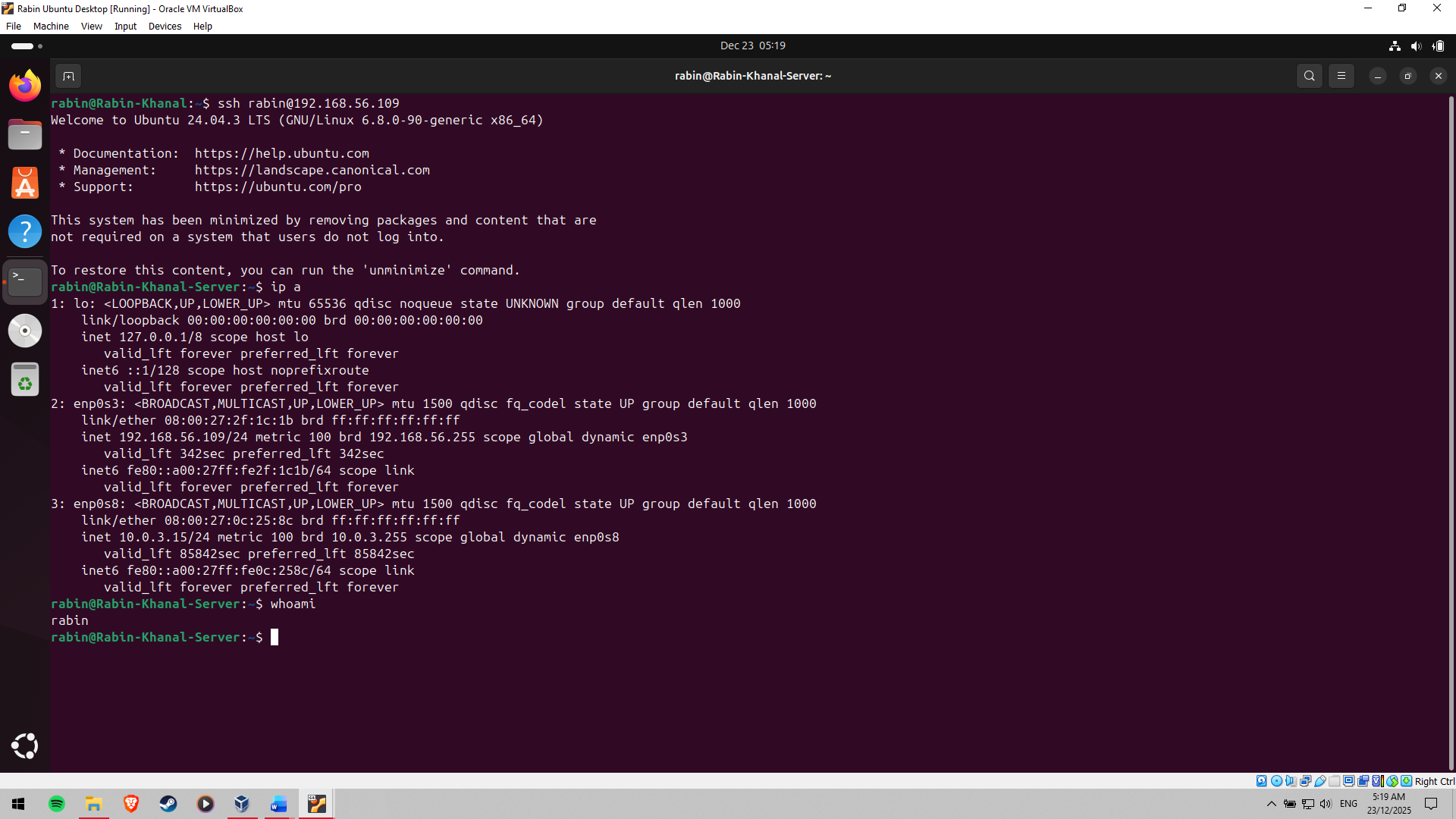


Figure 11: SSH to the server successful

# Phase 5: Advanced Security and Monitoring Infrastructure

Mandatory Access Control Implementation

In an effort to enhance system security over and above the conventional discretionary access regulations, Mandatory Access Control (MAC) is adopted through AppArmor to enforce it. Application behaviour is regulated by security policies imposed by AppArmor, wherein the access to system resources can be blocked even in cases where an application has been compromised, assuming that the policies comply with the base profile definitions. AppArmor is verified to be running and with policies enforced, as can be seen in Figure 12. AppArmor limits the potential effects of privilege abuse and unauthorised access by limiting the services to just the necessary permissions.

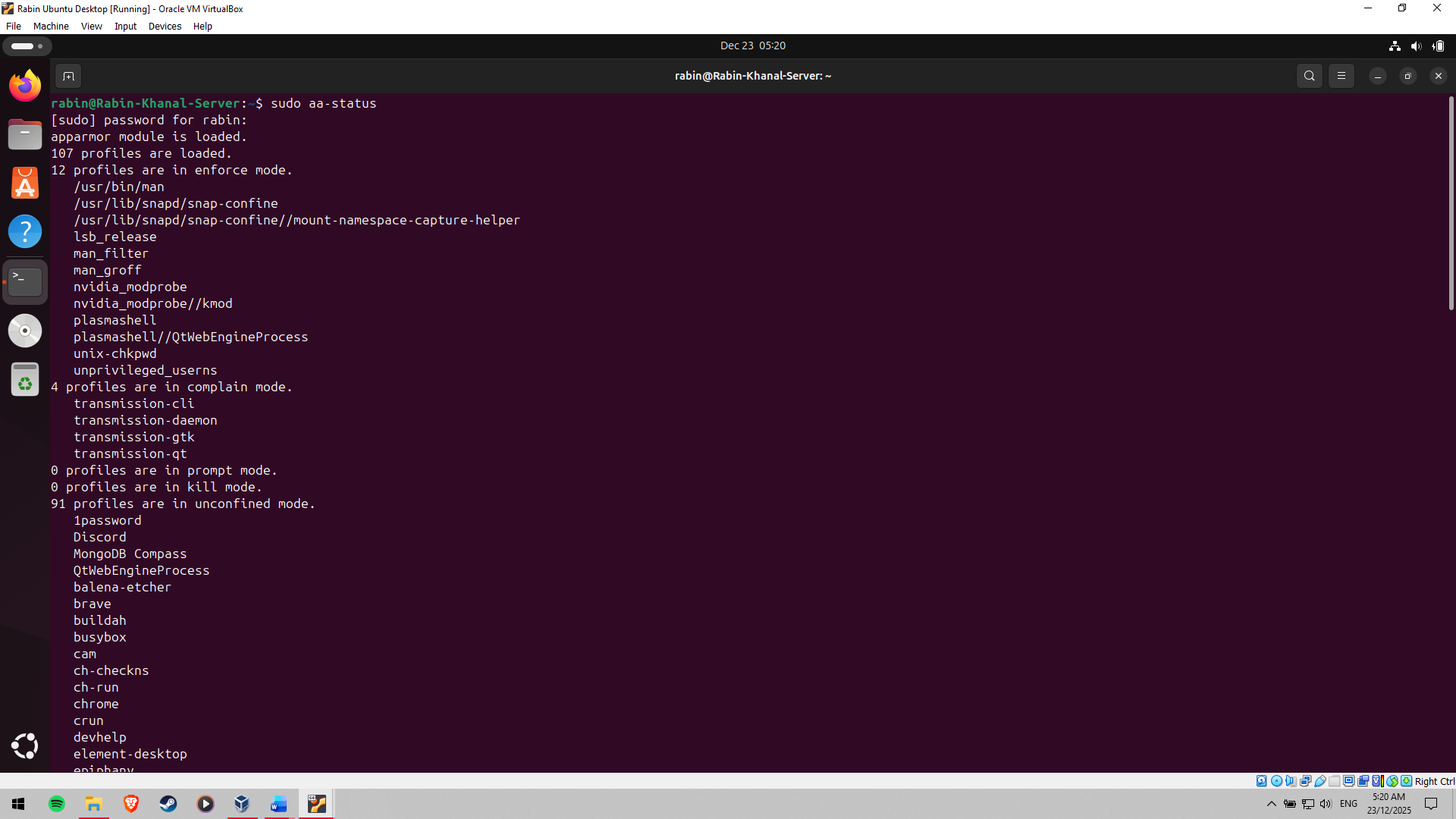


Figure 12: AppArmor Status on Server

**Auto Security Updates Configuration**

To make sure that important security fixes are done promptly without human intervention, the unattended-upgrades package is used to enable automatic security updates. This minimizes vulnerabilities that are known and promotes stability in the system. Configuration settings are checked to ensure that security updates are turned on and are working properly. Figure 13 can give testimony that unattended upgrades are working on the server, and this proves to be in line with the best practice of maintenance of security.

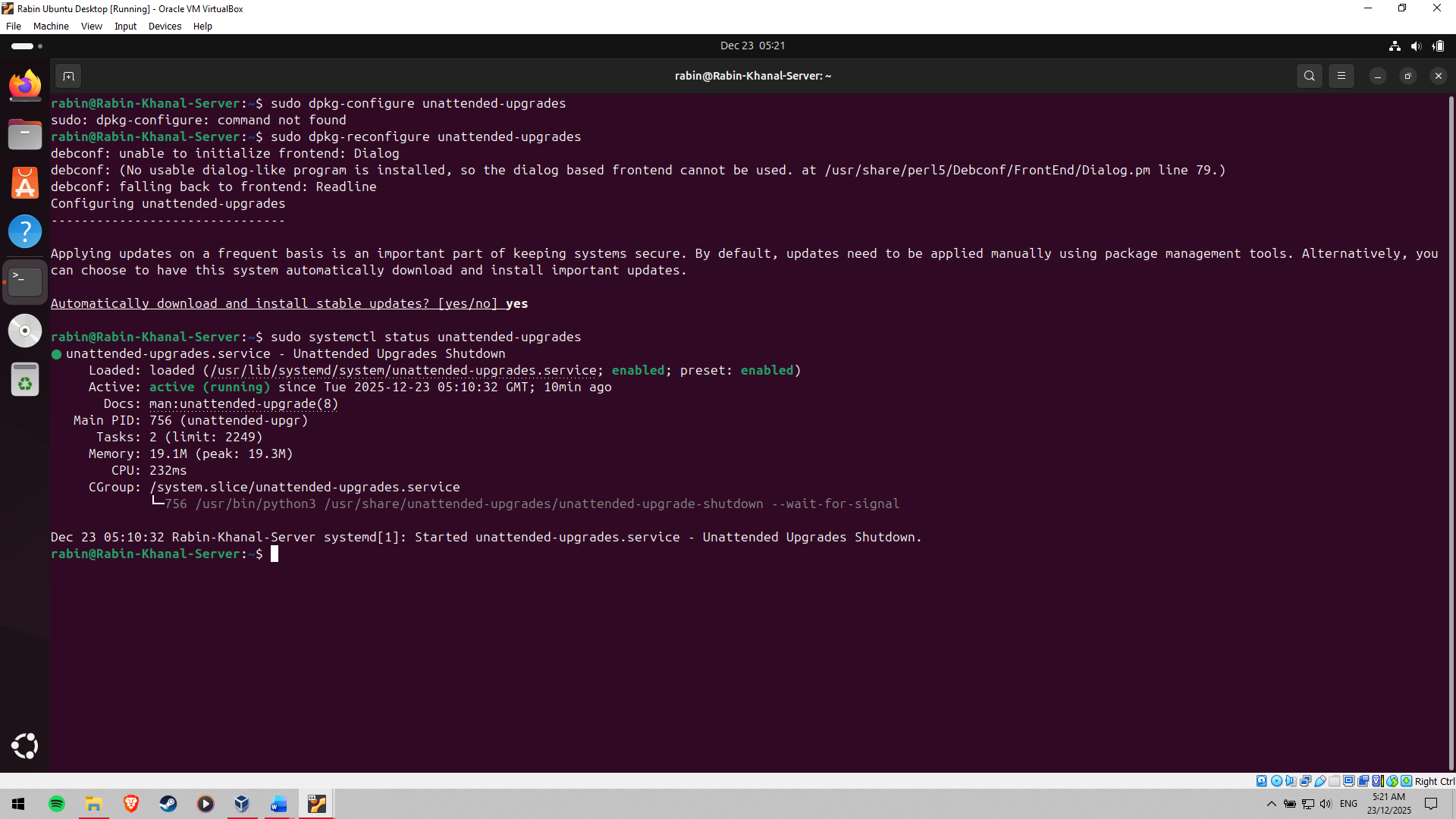


Figure 13: Unattended Upgrades enabled

**Intrusion Detection and Prevention with Fail2ban**

Fail2ban is implemented to improve the intrusion detection and prevention features, especially for SSH. It tracks the authentication records and automatically bans the IP addresses that show suspicious activity, like undergoing many failed attempts at logins. Fail2ban is set with the jailing rules that are specific to SSH to alleviate brute-force attacks [11]. The configuration to be used on SSH protection is illustrated in Figure 14, whereas Figure 15 indicates that the Fail2ban service is operational and is monitoring the system.

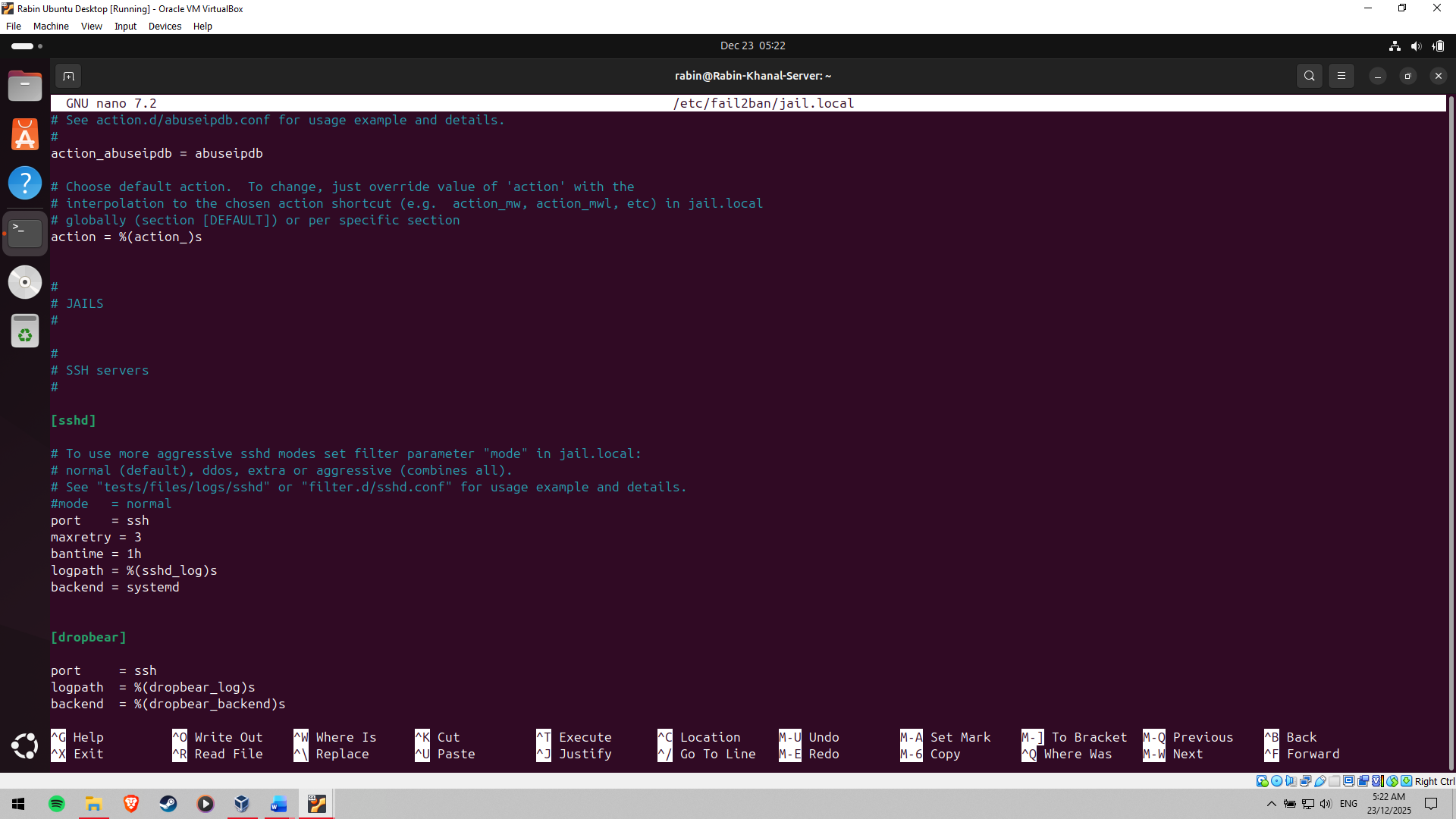


Figure 14: Fail2ban configured for SSH

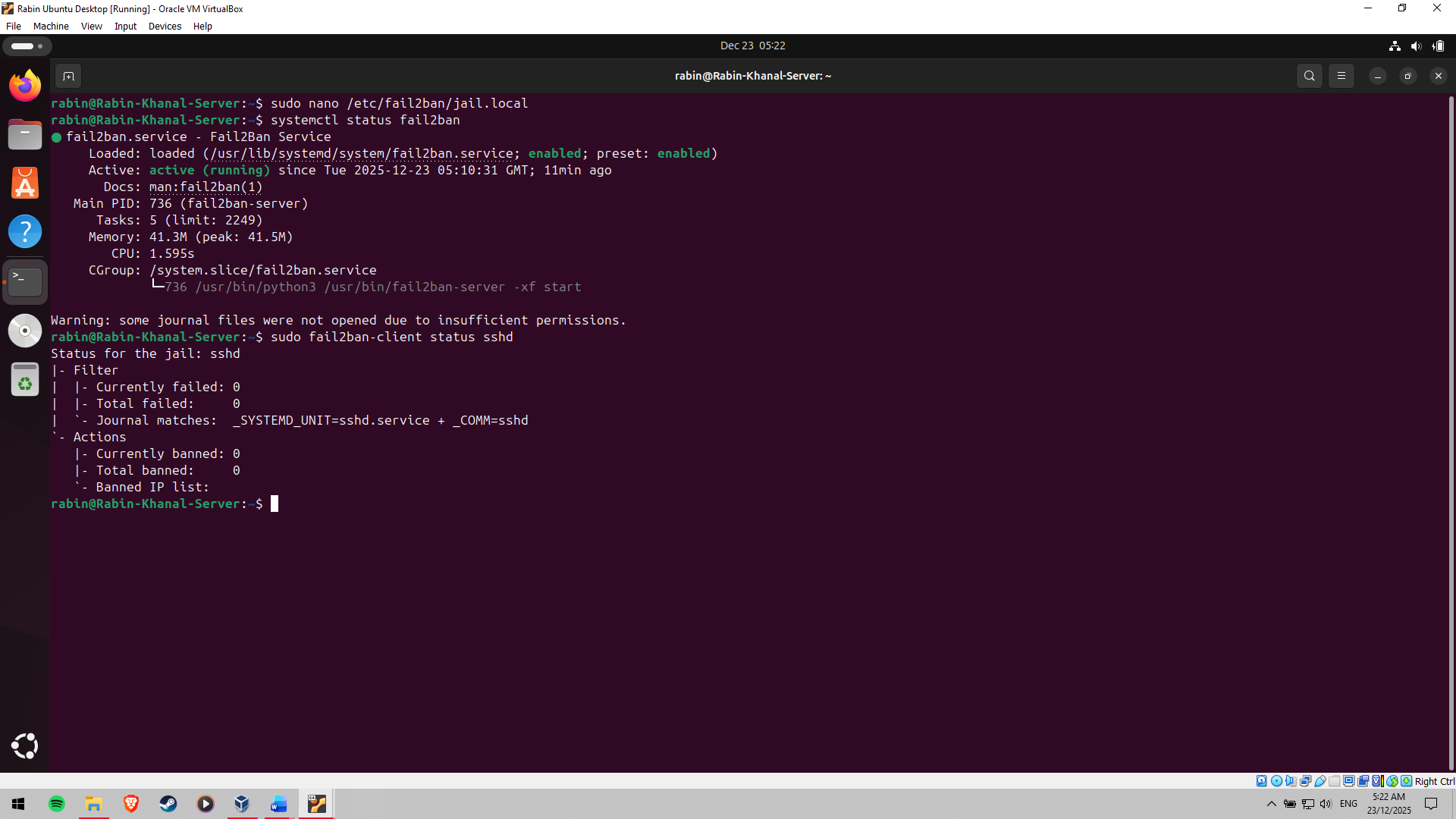


Figure 15: Fail2ban running on Server

**Security Baseline Verification Script**

A custom security-checking baseline script, security-baseline.sh, is created in order to automate the process of checking all security options done in Phases 4 and 5. The script verifies SSS hardening policies, firewall mode, AppArmor enactment, unable to progress upgrades and Fail2ban action. The script is separated into a series of lines, which are commented to give a definition of how it works to facilitate ease of understanding and ease of maintenance. The script code is shown in Figure 16, and the execution and the output of the successful execution of the baseline verification are shown in Figure 17.



Figure 16: Code for capturing security baseline

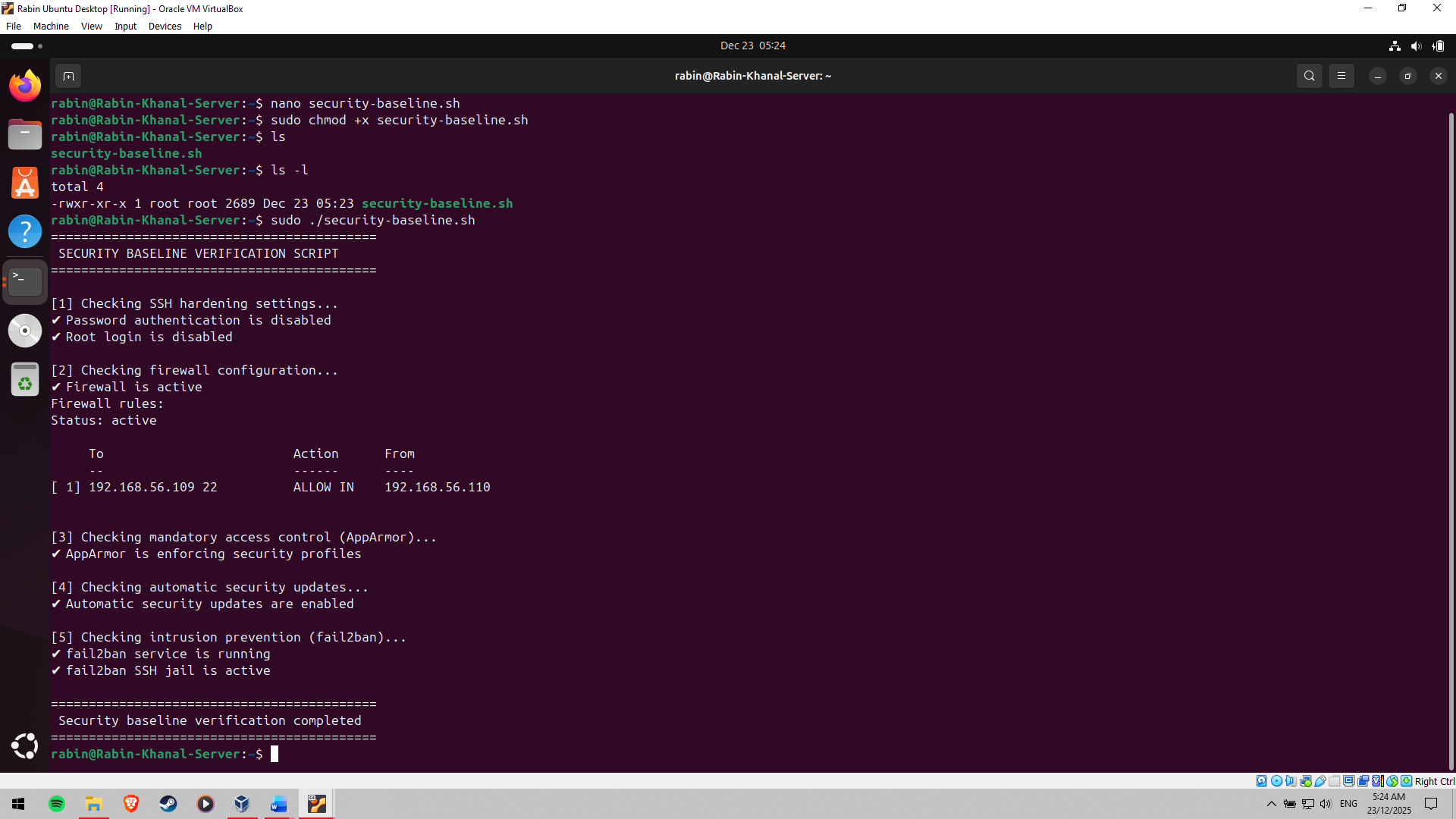


Figure 17: Security baseline measured

**Remote Monitoring Script Development**

In order to result in continuous performance monitoring, a remote monitoring script, monitor-server.sh, is developed on the workstation. This script links to the server over SSH and gathers key performance measurements like CPU activity, memory utilisation, disk transfer and network statistics. It has a script which is fully commented, and repeat execution is possible. The monitoring script is depicted in Figure 18, and the possibility of successful remote data acquisition of the server was verified in Figure 19.



Figure 18: Script for monitoring server

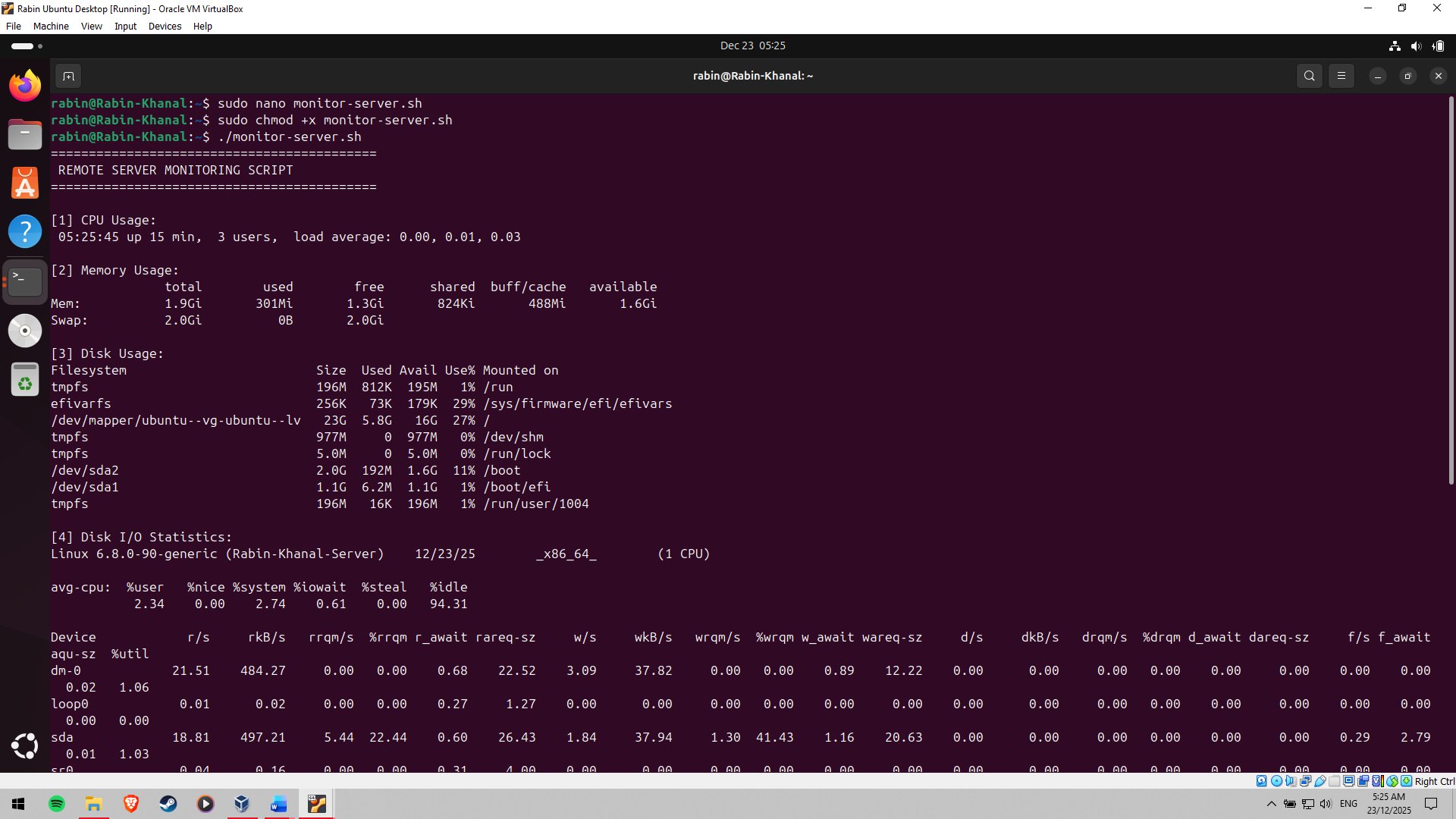


Figure 19: Remote Monitoring of the server completed

# Phase 6: Performance Evaluation and Analysis

**Performance Testing Methodology**

To test the behaviour of the Linux server under varying workload conditions, performance evaluation is performed according to a structured, repeatable methodology, and to analyse the behaviour of the server. Testing would start by performing baseline testing when the system is in an idle condition, just to determine a set of performance references. These baseline results are compared to the results of performance data that is collected at the application load testing and optimisation phases [12]. All the tests are launched and tracked remotely through SSH to ensure that compliance is also respected by the administration limitations of the assessment.

**Performance Analysis of CPU and Memory**

Test tools are used to evaluate CPU performance by generating a sustained load to the processor using stress testing tools. Figure 20 demonstrates a high CPU utilisation case, which enables monitoring the load averages, process scheduling, and responsiveness of the system under pressure. The memory performance can be analysed using the memory workloads that replicate the load that requires large segments of the system memory by simulating the RAM-intensive workloads.

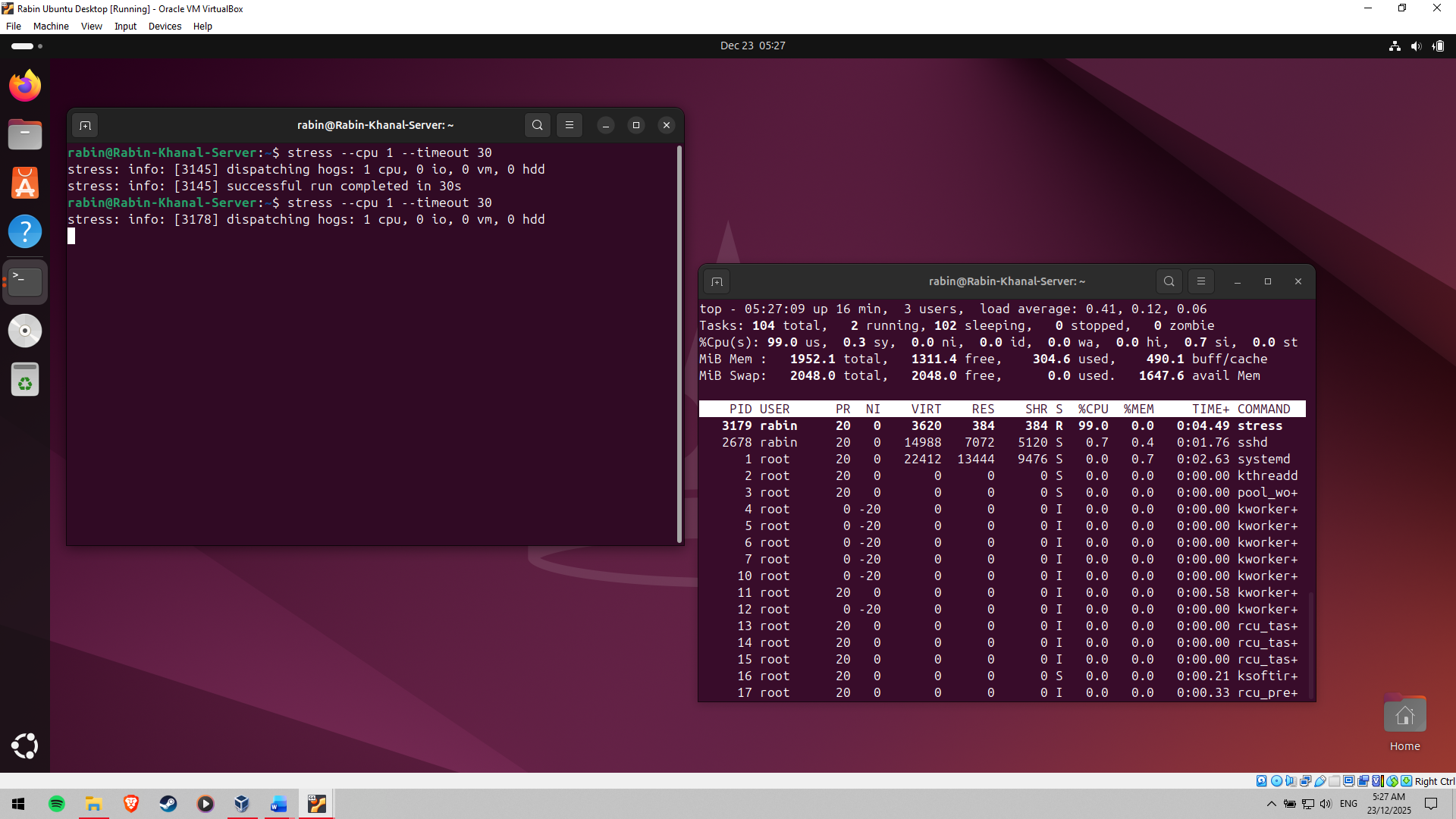


Figure 20: Stress Test for High CPU Usage

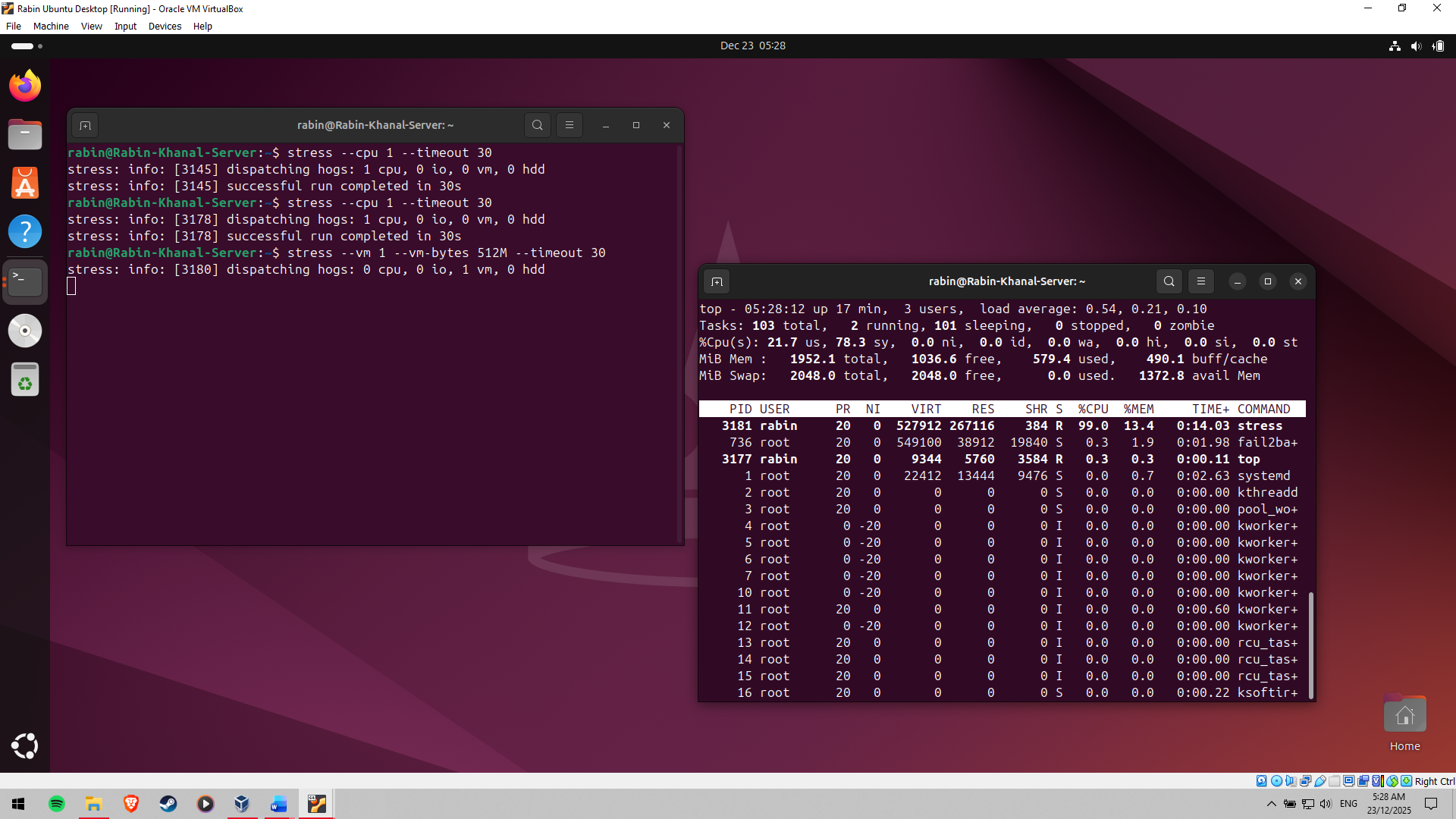


Figure 21: Stress Test for High RAM Usage

Figure 21 shows that high memory usage is observed when running stress testing, which allows assessing memory allocation efficiency and swap behaviour. These tests give us an idea of the efficiency of the operating system to handle the computational and memory resources in extreme situations.

**Network Performance Analysis and Disk I/O**

The benchmarking tools are used to determine disk I/O performance by creating random and sequential read/writestatements. Measurements of throughput, latency and I/O wait time are measured to determine any possible bottlenecks in the storage performance. Network testing utilities are used to determine network performance, including bandwidth, latency and packet loss between the workstation and the server. This analysis is necessary to guarantee that network communication is not a restrictive factor in service functioning.

**Bottleneck Identification and Optimisation**

The testing is done to examine performance data that can be used to detect bottlenecks in the system. Other typical bottlenecks are CPU saturation by high workload and higher memory pressure when performing operations that demand heavy use of RAM. In order to correct those problems, there are optimisation methods like tuning of the kernel parameters, process priority adjustment and fine-tuning of application settings. Post-optimisation testing is done in order to measure improvements. Findings indicate that there are quantifiable improvements in the stability of the systems, responsiveness, and resource usage available.

**Performance Visualisation and Evidence**

Collected performance data is systematised into tabular form and analysed using charts and graphs in order to aid in this analysis. These visualisations allow comparing the baseline, load and optimised state of performance. Logs, such as command logs and monitoring logs, are also captured in order to prove results and give support to analytical conclusions.

# Phase 7: Security Audit and System Evaluation

**System Security Audit with Lynis.**

On the Linux server, a full security audit is carried out with the help of the Lynis auditing tool to assess the general security state of the server. Lynis conducts a comprehensive evaluation of system settings, installed packages, kernel settings and security controls. Figure 22 is the report on the Lynis audit, where the levels of compliance are indicated, and possible points of improvement are provided. Advice given by Lynis is revisited to prove that the established controls in place are in line with the industry best practices.

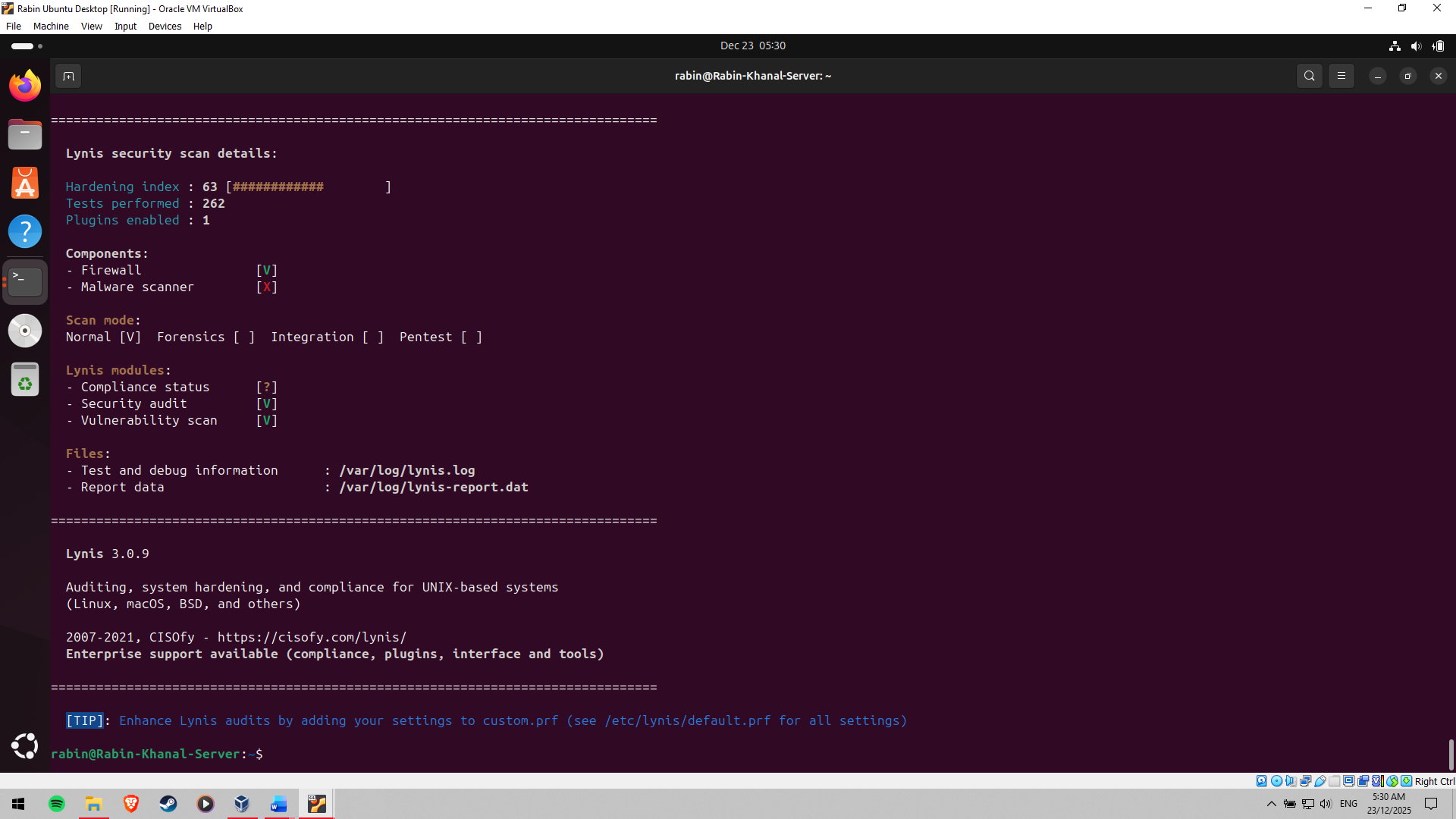


Figure 22: Lynis Audit of Server

**Network Security Checking with Nmap**

Nmap is used to determine the security at the network level to identify the services that are available and the open ports on the server. A TCP connect scan is conducted between the workstation, and it is done to ensure that only a service that has been explicitly allowed is available. The Nmap scan result is reported in Figure 23, where it is seen that SSH is the only externally available service and firewall regulations are operating as expected. This test will determine the efficiency of network access controls performed at previous stages.

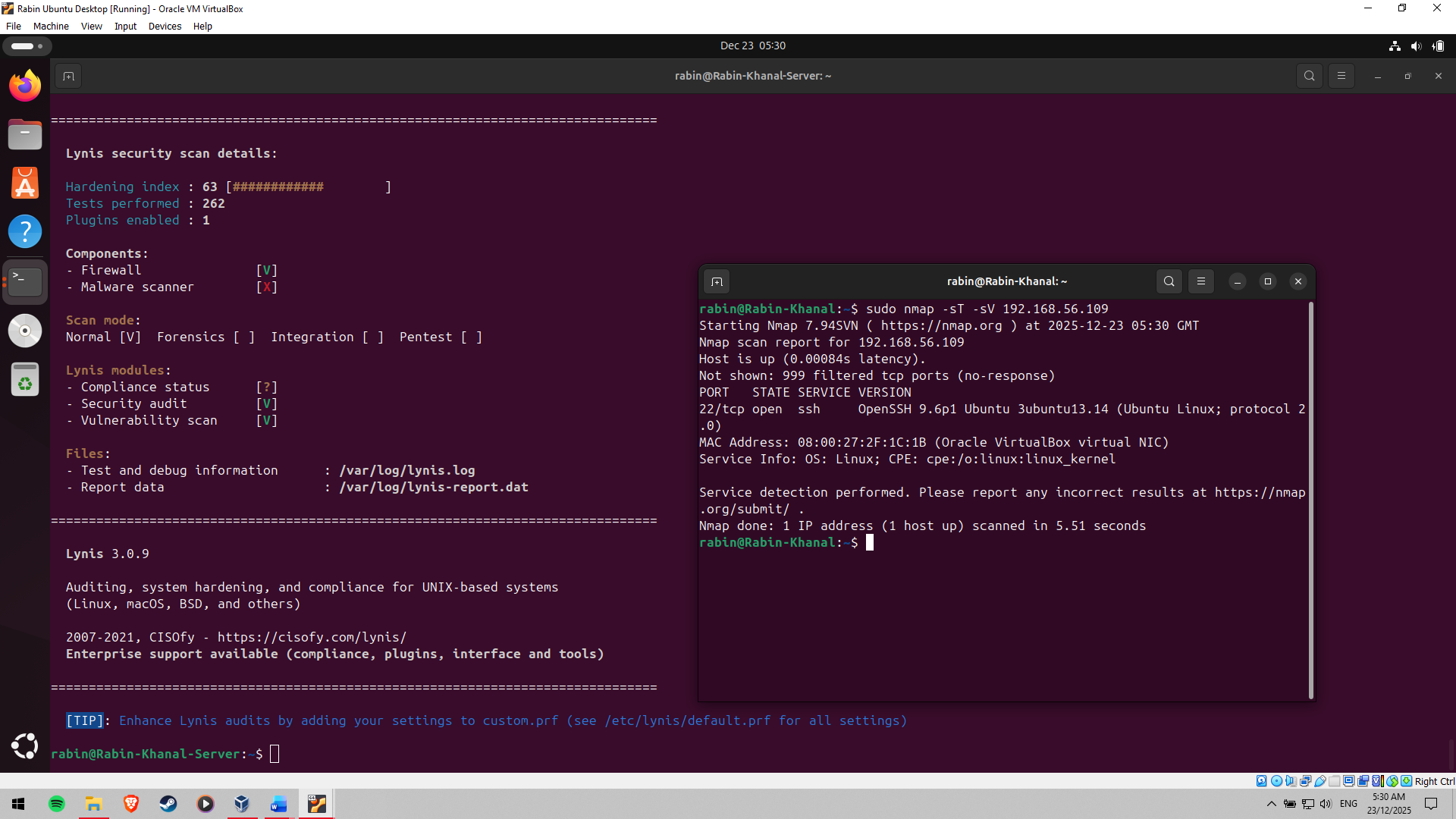


Figure 23: TCP connect scan using Nmap

**Access Control and Service Audit**

The need to check access controls is undertaken so that user permissions and mandatory access control policies are appropriately enforced. AppArmor profiles are also checked in order to ensure that critical services are isolated and are running within specified security specifications. Also, a service audit is performed to determine all the operating services in the server. Figure 24 provides a list of active services under which each of them is expressed and substantiated depending on the need for operation. Superfluous services are shut off or eliminated to decrease system attack targets.

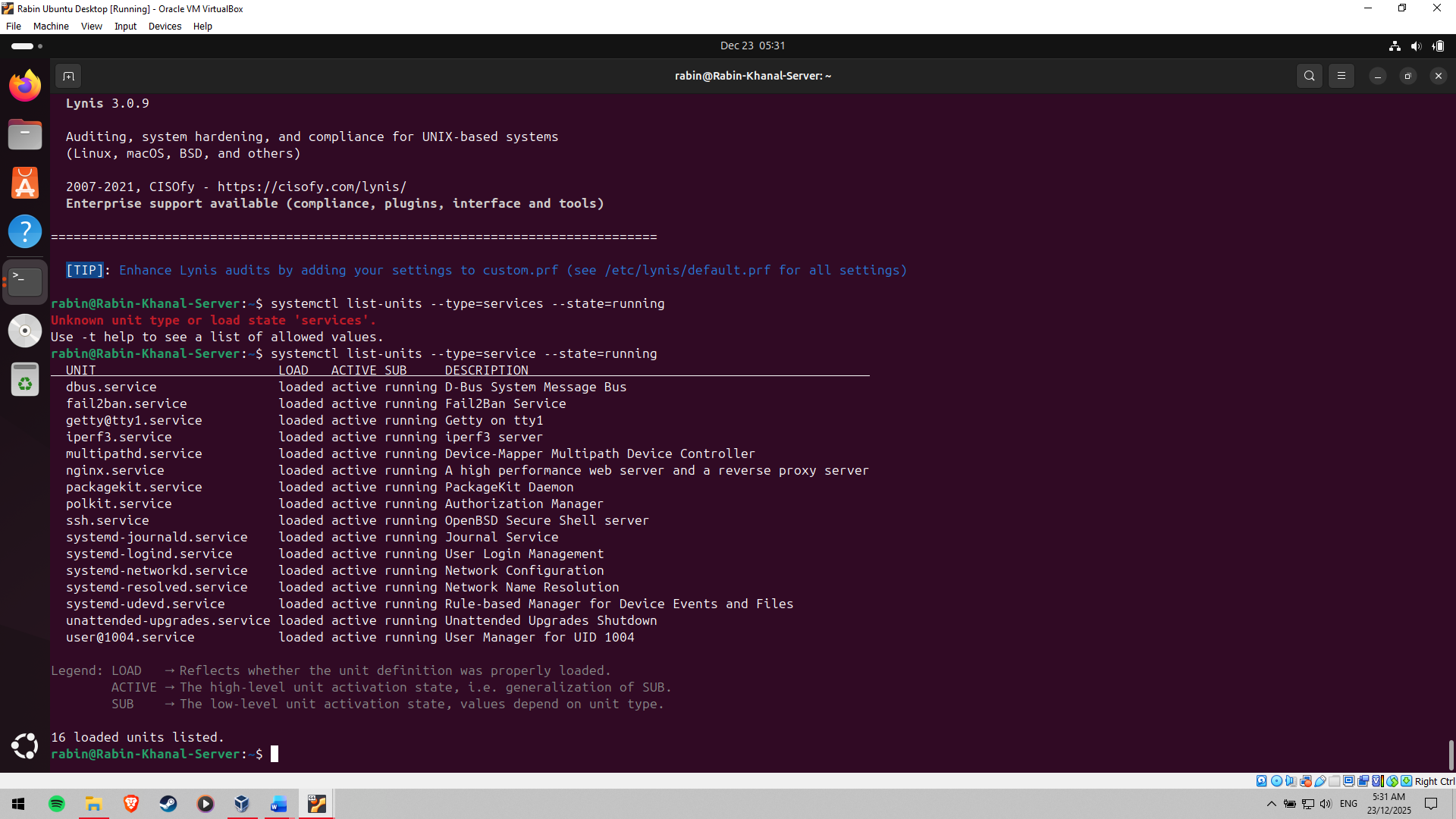


Figure 24: Services running on Linux Server

**Overall System Evaluation**

The results provided by the security audit and service review indicate that the system is well-configured and layered security controls have been used to minimise the risks. The server is in a high-security position and is not impacted by functional and performance issues, suggesting that it meets the aim of the test.

# Conclusion

This test illustrates the successful setup, security hardening and performance testing of a headless Linux server that is completely managed through SSH. The server had a sound security posture and had been deliberately configured with regular and enhanced security control features such as SSH hardening, firewall setup, AppArmor enforcement, automatic updates, and Fail2ban intrusion prevention. CPU, memory, disk I/O, and network workload performance testing revealed the possible bottlenecks, which were then countered using specific optimisations. Extensive monitoring, auditing backed up by scripts and visualised performance data, gave quantifiable results in the behaviour of the system under different conditions. In general, the project emphasises the extreme relevance of safe and optimal Linux server management in current computing systems and strengthens the command-line expertise practicability.

References

[1] P. Jakić, A. Elsadai, and M. Tair, “Comparative Analysis of the Impact of Server Operating Systems on Web Site Performance,” in *Proceedings of the International Scientific Conference - Sinteza 2021*, Beograd, Serbia: Singidunum University, 2021, pp. 180–186. <https://www.researchgate.net/publication/352888701_Comparative_Analysis_of_the_Impact_of_Server_Operating_Systems_on_Web_Site_Performance>

[2] V. Tiwari, S. Upadhyay, J. K. Goswami, and S. Agrawal, “Analytical Evaluation of Web Performance Testing Tools: Apache JMeter and SoapUI,” in *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE, Apr. 2023, pp. 519–523.

<https://www.researchgate.net/publication/371226310_Analytical_Evaluation_of_Web_Performance_Testing_Tools_Apache_JMeter_and_SoapUI>

[3] P. Chatterjee, R. Mukherjee, I. Das, and T. Roy, “REAL-TIME PERFORMANCE BENCHMARKING USING VMSTAT/IOSTAT,” 2025, [Online]. Available: [https://www.researchgate.net/publication/393509597\_REAL-TIME\_PERFORMANCE\_BENCHMARKING\_USING\_VMSTATIOSTAT](https://www.researchgate.net/publication/393509597_REAL-TIME_PERFORMANCE_BENCHMARKING_USING_VMSTATIOSTAT%20)

[4] A. B. Guide, “A Beginners Guide to Linux firewall,” 2021, [Online]. Available: <https://safe.security/wp-content/uploads/a-beginners-guide-to-linux-firewall.pdf>

[5] T. Ecarot, S. Dussault, A. Souid, L. Lavoie, and J.-F. Ethier, “AppArmor For Health Data Access Control: Assessing Risks and Benefits,” in *2020 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, IEEE, Dec. 2020, pp. 1–7. <https://www.researchgate.net/publication/348997066_AppArmor_For_Health_Data_Access_Control_Assessing_Risks_and_Benefits>

[6] G. Thiyagarajan, V. Bist, and P. Nayak, “The Hidden Dangers of Outdated Software: A Cyber Security Perspective,” 2025, [Online]. Available: <https://www.researchgate.net/publication/391910669_The_Hidden_Dangers_of_Outdated_Software_A_Cyber_Security_Perspective>

[7] G. Martinovic, J. Balen, and B. Cukic, “Performance Evaluation of Recent Windows Operating Systems,” 2012, [Online]. Available: [https://www.researchgate.net/publication/260811286\_Performance\_Evaluation\_of\_Recent\_Windows\_Operating\_Systems](https://www.researchgate.net/publication/260811286_Performance_Evaluation_of_Recent_Windows_Operating_Systems%20)

[8] L. Gao, Z. Luo, and L. Wang, “Convolutional Neural Network Acceleration Techniques Based on FPGA Platforms: Principles, Methods, and Challenges,” *Information*, vol. 16, no. 10, p. 914, Oct. 2025, [https://www.mdpi.com/2078-2489/16/10/914](%20https:/www.mdpi.com/2078-2489/16/10/914)

[9] D. Gudu, M. Hardt, L. Brocke, and G. Zachmann, “Enabling Secure Shell Access with OpenID Connect,” *Comput. Softw. Big Sci.*, vol. 9, no. 1, p. 5, Dec. 2025, <https://www.researchgate.net/publication/390468116_Enabling_Secure_Shell_Access_with_OpenID_Connect>

[10] M. Laakso, “Study of the Nordic SSH Journal Publishing Landscape: A Report for the Nordic Publications Committee for Humanities and Social Science Periodicals (NOP-HS),” 2021, [Online]. Available: [https://nos-hs.org/wp-content/uploads/2021/12/study-of-the-nordic-ssh-journal-publishing-landscape.pdf](https://nos-hs.org/wp-content/uploads/2021/12/study-of-the-nordic-ssh-journal-publishing-landscape.pdf%20)

[11] B. Wibowo and L. Hafiz, “Risk Analysis of Bruteforce Attacks on Webserver with Telegram Notifications,” *J. Komput. dan Elektro Sains*, vol. 3, no. 1, pp. 28–32, Jan. 2025, <https://www.researchgate.net/publication/387934932_Risk_Analysis_of_Bruteforce_Attacks_on_Webserver_with_Telegram_Notifications>

[12] M. Boras, J. Balen, and K. Vdovjak, “Performance Evaluation of Linux Operating Systems,” in *2020 International Conference on Smart Systems and Technologies (SST)*, IEEE, Oct. 2020, pp. 115–120. <https://www.researchgate.net/publication/347189472_Performance_Evaluation_of_Linux_Operating_Systems>