**Configuration, Security Hardening, and Performance Evaluation of a Headless Linux Server Using Remote Command-Line Administration**

Table of Contents

[Introduction 2](#_Toc217378776)

[Phase 1: System Planning and Distribution Selection 2](#_Toc217378777)

[Phase 2: Security Planning and Testing Methodology 4](#_Toc217378778)

[Phase 3: Application Selection for Performance Testing 5](#_Toc217378779)

[Phase 4: Initial System Configuration & Security Implementation 7](#_Toc217378780)

[Phase 5: Advanced Security and Monitoring Infrastructure 11](#_Toc217378781)

[Phase 6: Performance Evaluation and Analysis 16](#_Toc217378782)

[Phase 7: Security Audit and System Evaluation 18](#_Toc217378783)

[Conclusion and Reflection 21](#_Toc217378784)

[References 22](#_Toc217378785)

# **Introduction**

This assignment is dedicated to the configuration, security and performance assessment of the Linux operating system that is a server. The assessment is done through a headless Ubuntu server and remote SSH to another workstation, where the user builds hands-on skills in command-line proficiency and professional system administration. The research has emphasized the significance of optimization configuration of operating systems, which may in turn greatly cut down resource usage in a data centre, which is a factor in energy conservation.

This assessment will be designed as phased activities in system planning, security implementation, performance testing on the system based on the applications, and auditing of the system. The phases include installation, securing, and testing of the server and performance assessment of various workloads. Through the integration of security hardening and analysis of the performance, the project characterizes best practices in the management of the Linux system by using real-world and production-like configurations.

# **Phase 1: System Planning and Distribution Selection**

**Overview: System architecture**

This assessment has a dual-system model architecture comprising a Linux server and a remote workstation to administer the system. The server is running Ubuntu Server in a headless version, that is, there is no graphical desktop system installed. This design option requires the use of only command-line tools and resembles the actual server environment in the world that is typical of data centres and enterprise environments.

Both systems are run as a virtual machine in Oracle VirtualBox, which permits experiment control, isolation and repeatability. The virtual machines will be set up in the same virtual network, which allows them to communicate directly without the need to share the administration and service provision roles. Connectivity was successfully implemented between the two systems, and it was reported that the network was set up correctly and the remote access was functional.

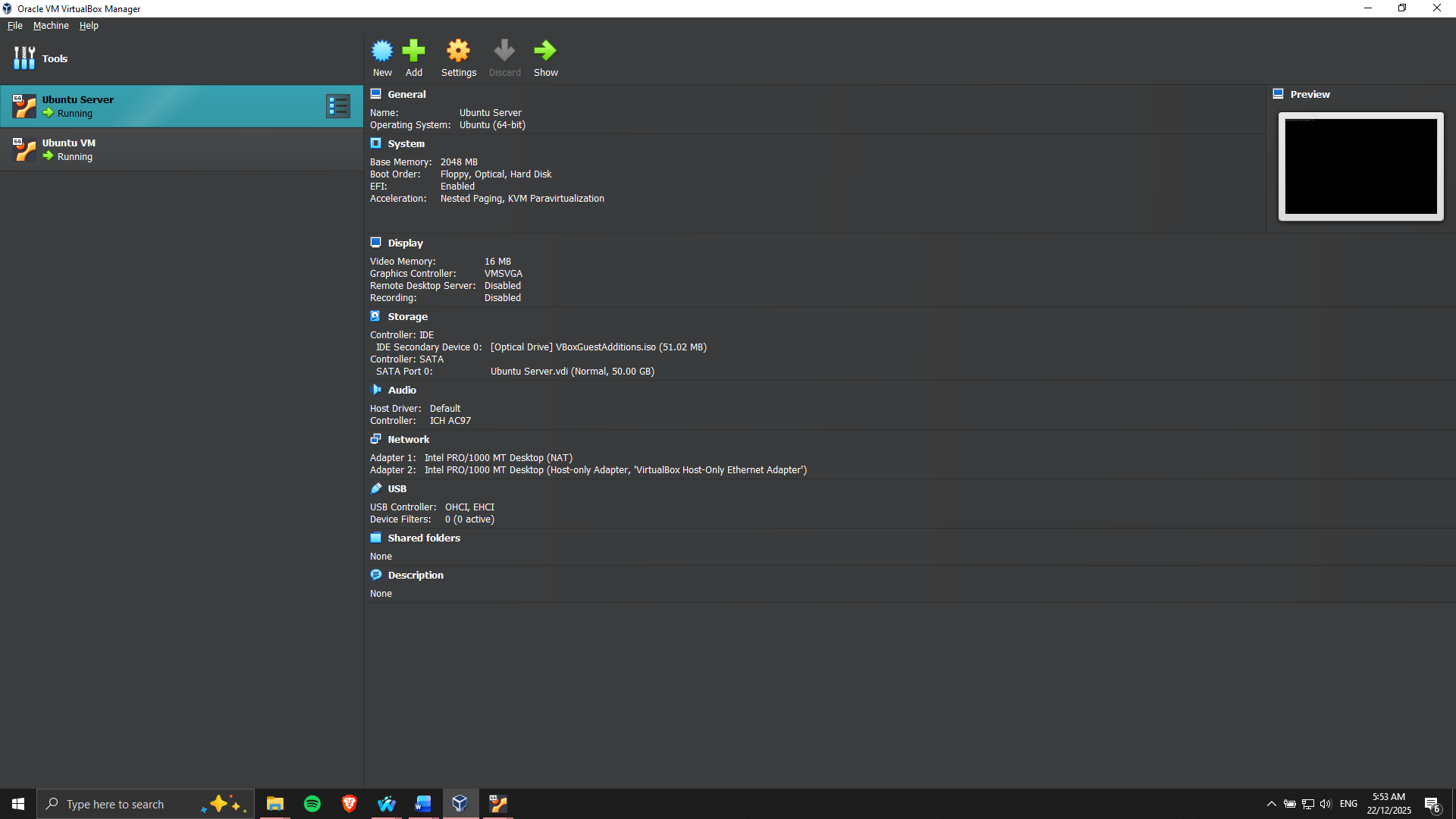


Figure 1: Ubuntu Server and VM running

**Selection and Justification of Distribution**

Ubuntu Server was chosen as a server operating system because it has long-term support (LTS), stability and support of the community and the enterprise. Ubuntu Server is an appropriate rebuttal between the latest development in software packages and the extended security features, so it can be used in both learning and production-like settings, as compared to other distributions like CentOS Stream or Debian. It is also well documented and can be used with security software, including UFW, AppArmor and fail2ban, which also makes it an option of use [1].

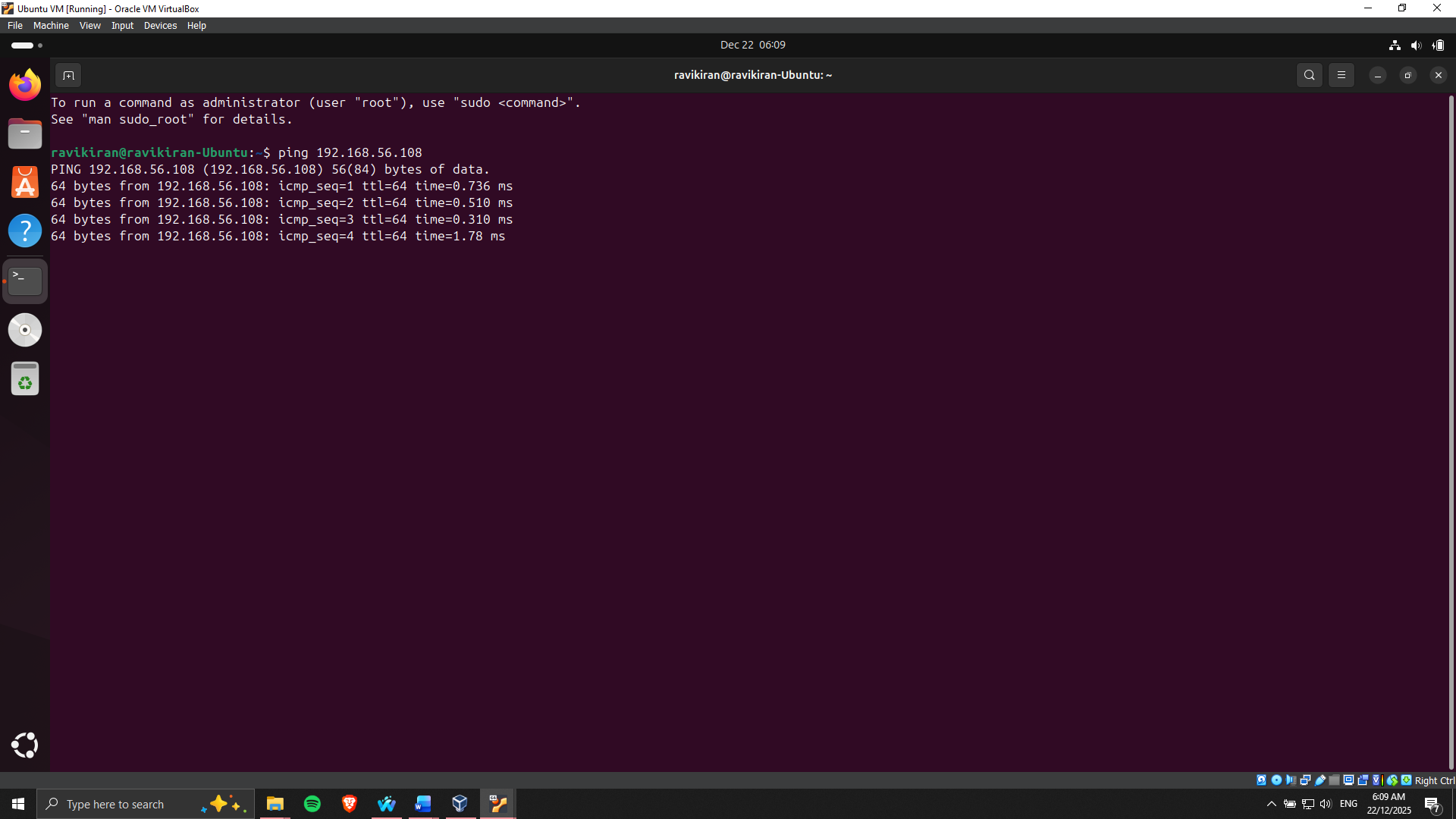


Figure 2: Connection between VMs successful

**Workstation/ Network configuration**

The workstation system offers the SSH client environment, which can be a Linux desktop virtual machine or a host operating system. The configuration of a network on VirtualBox is through a private network configuration, which provides secure communication between the workstation and the server. The IP addressing was recorded with the help of the command-line tools and ensured the configuration of the network interfaces and the architecture of the underlying system.

# **Phase 2: Security Planning and Testing Methodology**

**Security Planning Approach**

Systematic security planning was done in order to have an outline plan of a secure baseline on the Linux server prior to performing performance testing. The main purpose was to reduce the attack surface of the system; however, ensuring remote administration was possible. The set of security controls was chosen with a view to the common threat vectors against publicly accessible Linux servers, such as unauthorised access, brute-force and misconfiguration vulnerability.

Combative security actions that were to be undertaken included hardening Secure Shell (SSH), firewall, user privilege control, and automatic access management, mandated security updates and detecting intrusions. The measures follow the best practices in the industry and establish a platform for the safe running of the system.

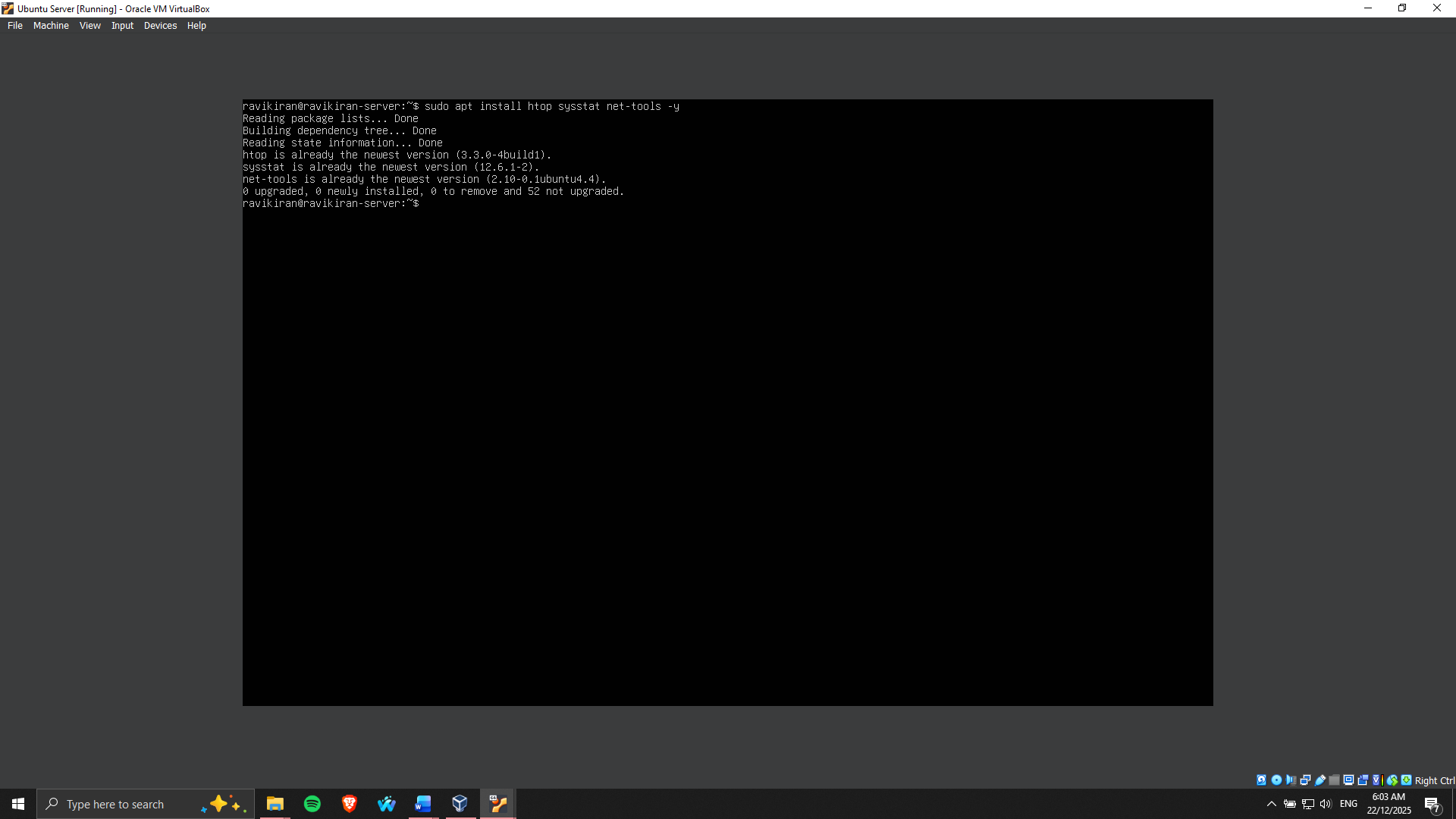


Figure 3: Tools installed for monitoring

Figure 3 shows that the monitoring and performance analysis tools are installed in the server, which proves preparedness to further performance evaluation and monitoring actions remotely.

**Performance Testing Methodology**

The performance testing role was created to test the behaviour of the systems with varying workloads, while keeping the systems secure and providing remote access. The surveillance was performed over the workstation by means of SSH-based tools in order to prevent adding local overhead to the server. The key performance metrics that were chosen to be analysed were CPU utilisation, memory usage, disk input/output operations, network throughput and latency in the entire system [2].

Baseline measurements were going to be used to record the normal system behaviour before load testing. This was succeeded by the controlled application-level stress testing in order to mimic intensive CPU-based workloads and intensive memory-based workloads. Information gathered in the testing was organised to be compared and thus find out the bottlenecks in the performance and assess the optimisation plans [3].

# **Phase 3: Application Selection for Performance Testing**

**The workload Classification and Selection Criteria**

Various applications which represent the various workloads types were determined to ascertain the performance nature in the Linux server under different operating conditions. The main purpose was to tune the operating system to the amount of required load, to evaluate the use of system resources, including CPU, memory, disk I/O, as well as network bandwidth. The applications have been chosen based on the ability of generating predictable and measurable workloads and suitability of the application to execute in the headless server world using SSH.

The workload categories that were identified to be tested were CPU-intensive, memory-intensive, disk I/O intensive and network-intensive workloads. This method guarantees overall coverage of important components of the system and makes the performance comparison of different usage situations meaningful.

**Applications and Justification Selected**

In the case of CPU-intensive testing, a stress-testing tool was chosen to represent the maintained load on the processor over a number of cores. This allows the scheduling behavior of cpu, load capacity, and thermal performance to be observed during heavy computation. Workloads with a high memory usage were created with the use of such a tool that can allocate and consume a large amount of RAM to examine the memory usage pattern, caching behavior, and possible swapping.

Disk I/O performance measurements were performed based on file generation and read/write to measure the throughput and latency. Network-intensive testing entailed tools that are meant to create and measure network traffic so that bandwidth usage and latency can be assessed between the workstation and the server.

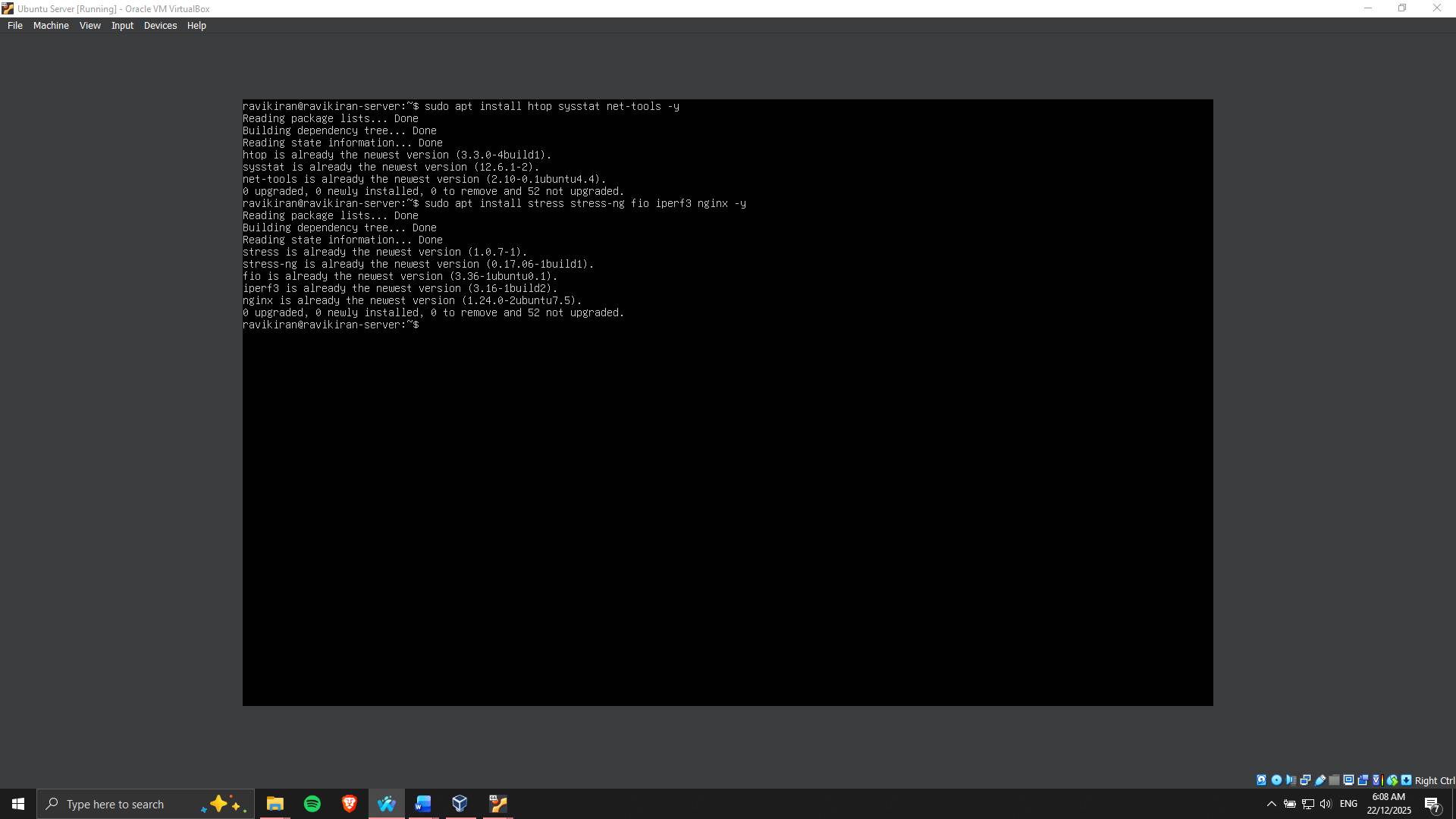


Figure 4: Application installed for performance testing

Figure 4 shows that some of the chosen performance testing applications have been successfully installed in the Linux server, which proves that it is ready to proceed with the further phases of testing.

**Expected Resource Profiles and Strategy of Monitoring**

Every application was assigned a profile of the usage of the resources, which could be monitored selectively during the execution process. The real-time metrics monitoring was captured with system monitoring tools by connecting to the servers remotely with the use of SSH, which did not cause much load to the servers. This is a well-organised selection and monitoring plan, which offers a secure background to elaborate performance assessment and optimisation during subsequent stages.

# **Phase 4: Initial System Configuration & Security Implementation**

**Secure Remote Access User**

The first system configuration was concerned with implementing safe and restricted remote administrative access into the Linux server. To ensure that the management was done remotely through the assigned workstation, the secure shell (SSH) was set up as the only mode of administration. To prevent vulnerabilities to brutality and credential-based attacks, public key-based authentication was introduced to substitute the password-based authentication, which made the attacks significantly harder [4]. The SSH public key of the workstation was safely transferred to the server and inserted into the authorized keys file as shown in Figure 5 and Figure 7.

In order to further enhance the security of SSH, the root login and password verification were precisely turned off in the SSS daemon environment file. This denies direct root access and implements the principle of least privilege, requiring that an administrator log in as a non-root user and only escalate their privileges when it is reasonable. These insecure modes of authentication have been disabled, as shown in Figure 6.

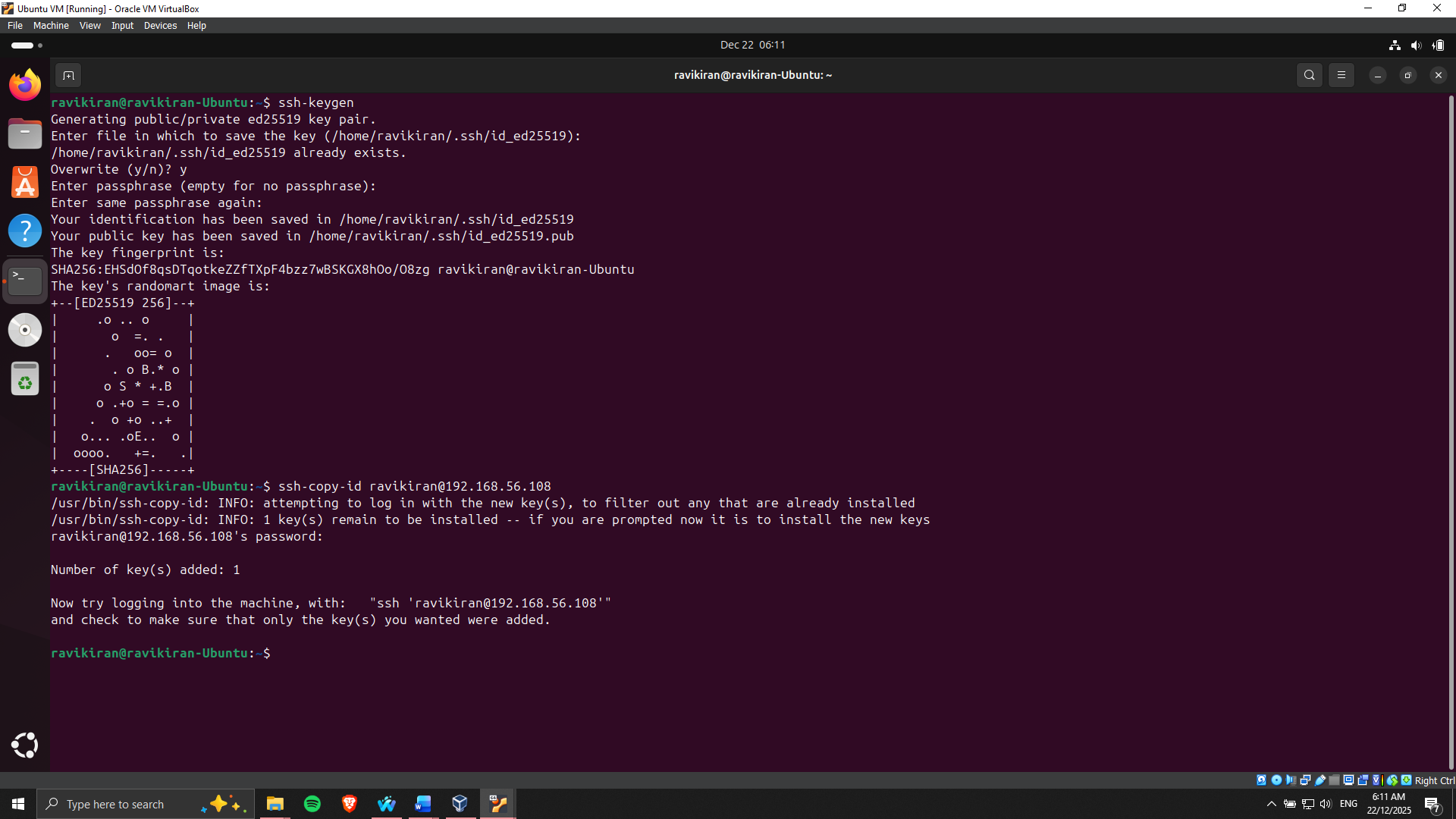


Figure 5: SSH key for the Desktop copied to the Server

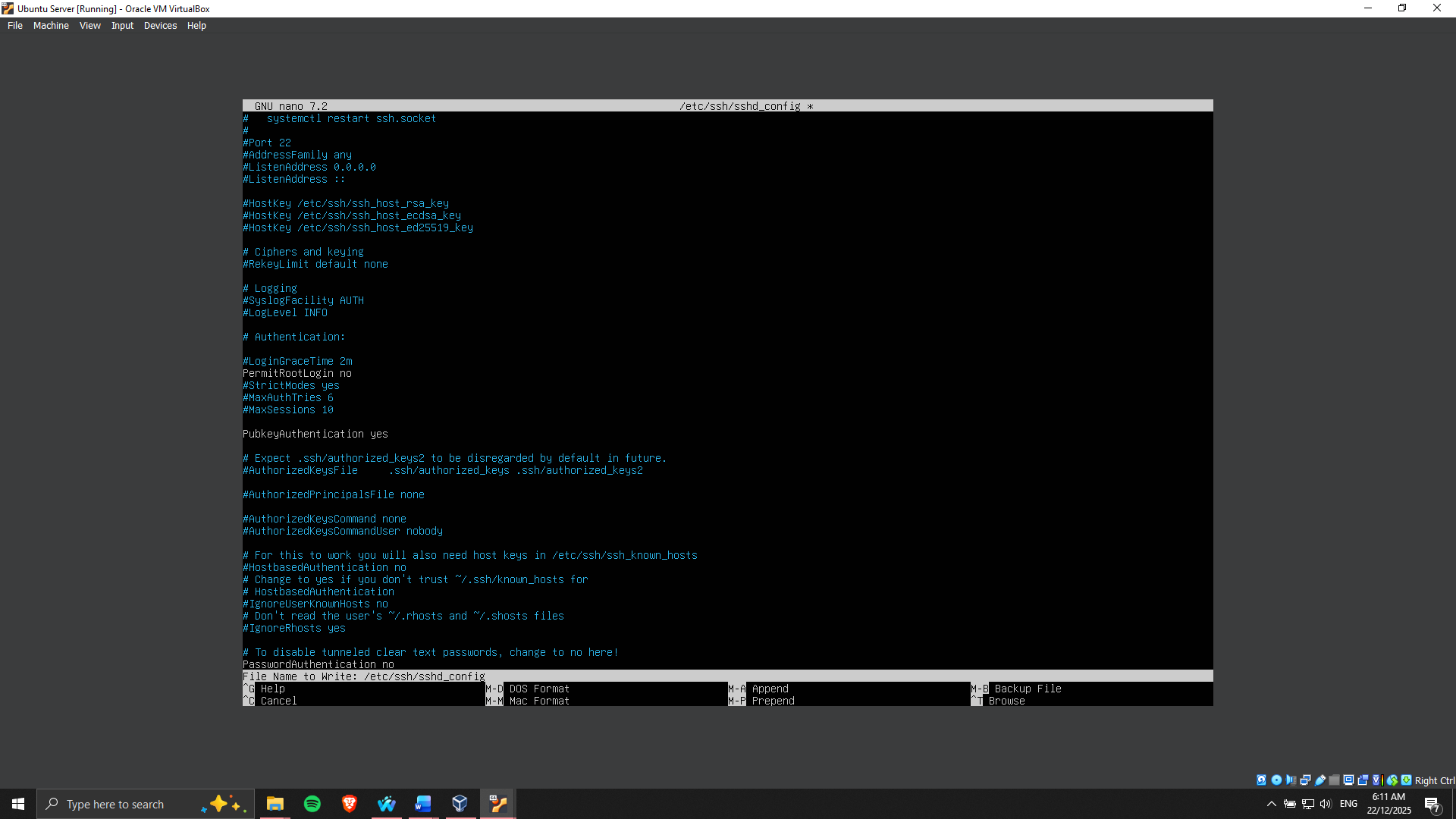


Figure 6: Root Login and Password authentication disabled

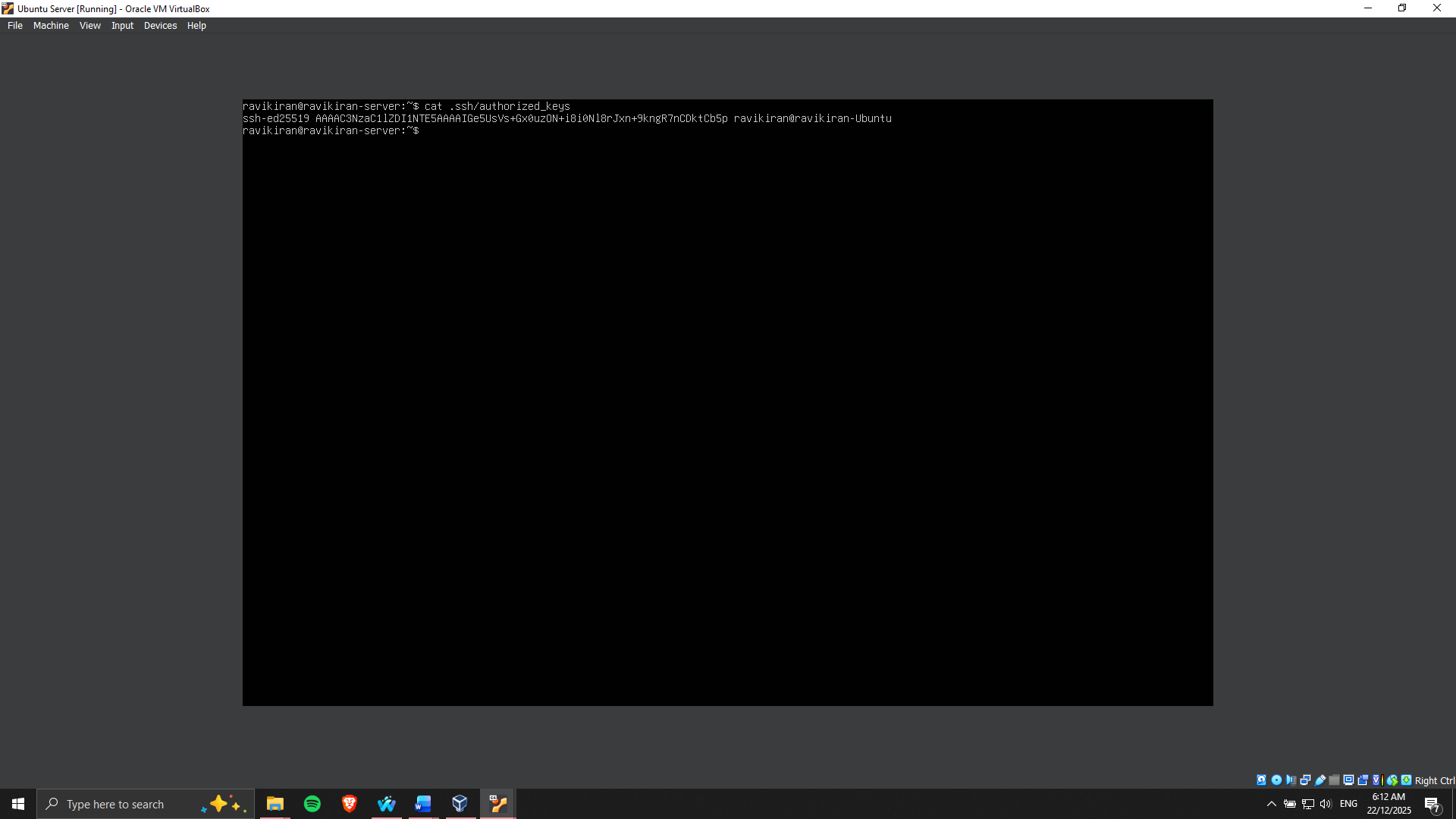


Figure 7: Authorised key present in the Server

**User and Privilege Management**

An administrative account was made to undertake all system administration activities; however, this was a non-root user. This user was provided with sudo privileges, which meant that they were allowed to execute administrative commands with controlled privileges and add accountability by recording the command. Limited use of the root account was imposed so that systemwide changes due to accidents or malicious intent were reduced. Such a practice is best practice for Linux server management and improves the auditability of the system.

**Firewall Grouping and Network Access Control**

Uncomplicated Firewall (UFW) was used to set up a host-based firewall to limit access to the server over the network. Firewall ruleset was configured to receive SSH connection only with the definite IP address of the workstation system, and this acted as a type of restriction to the outside world to access information of a known source [5]. The default had rejected all other inbound connections, minimising the attack surface exposure on the server. It can be seen in Figure 8 that UFW indeed was enabled, and it was actively applying the set rules.

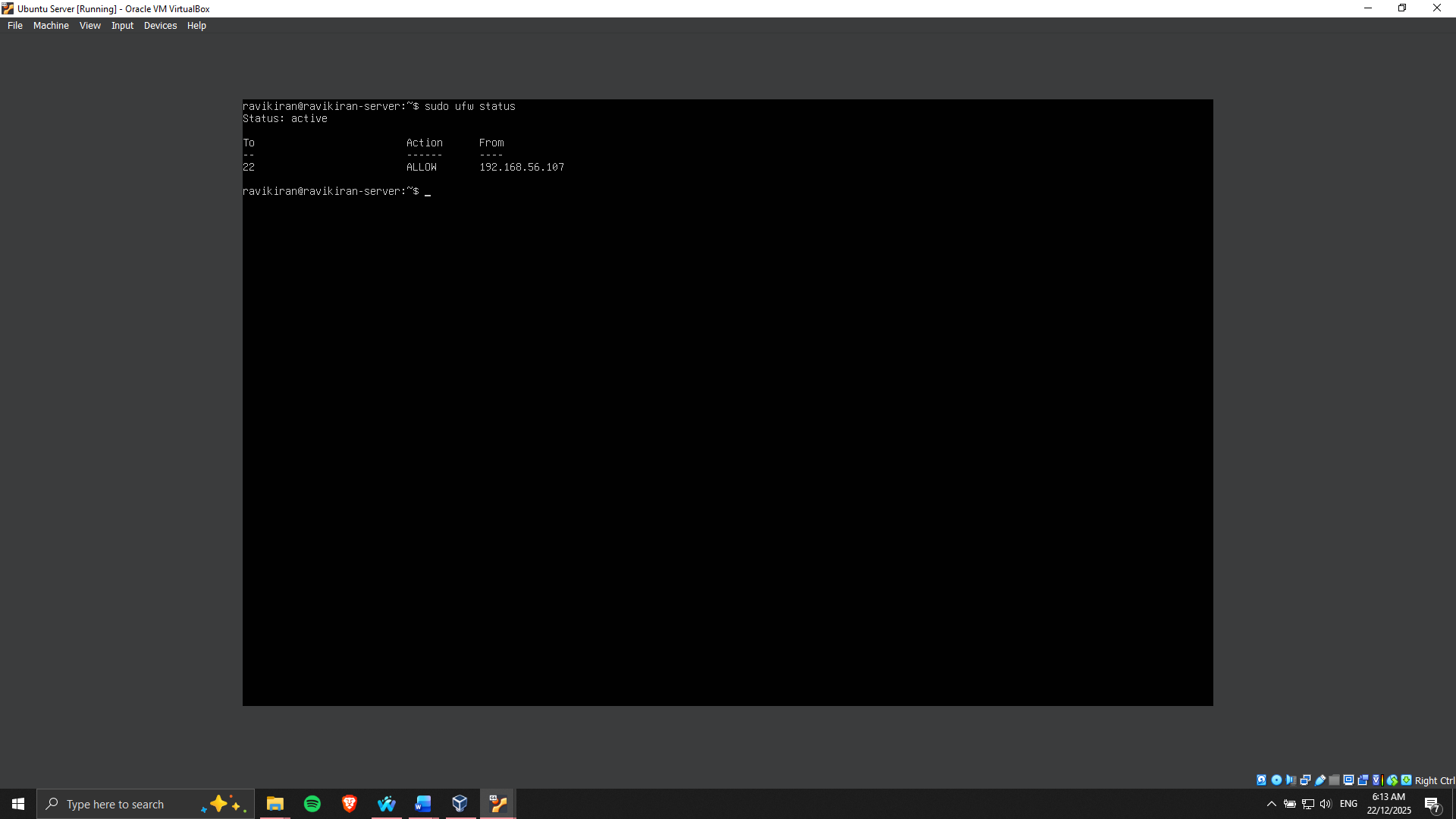


Figure 8: UFW active to allow SSH connection to the server

**Verification and remote administration Evidence**

The remote administration was proved successful by connecting with the server through the SSH command with the key-based authentication that was configured, as will be demonstrated in Figure 9. The entire configuration was done through SSH, within the limits of the assessment, as required. This step provided a safe, low-exposure server environment, which was a sound base of sophisticated protection controls and acceptance control testing in the stages [6].

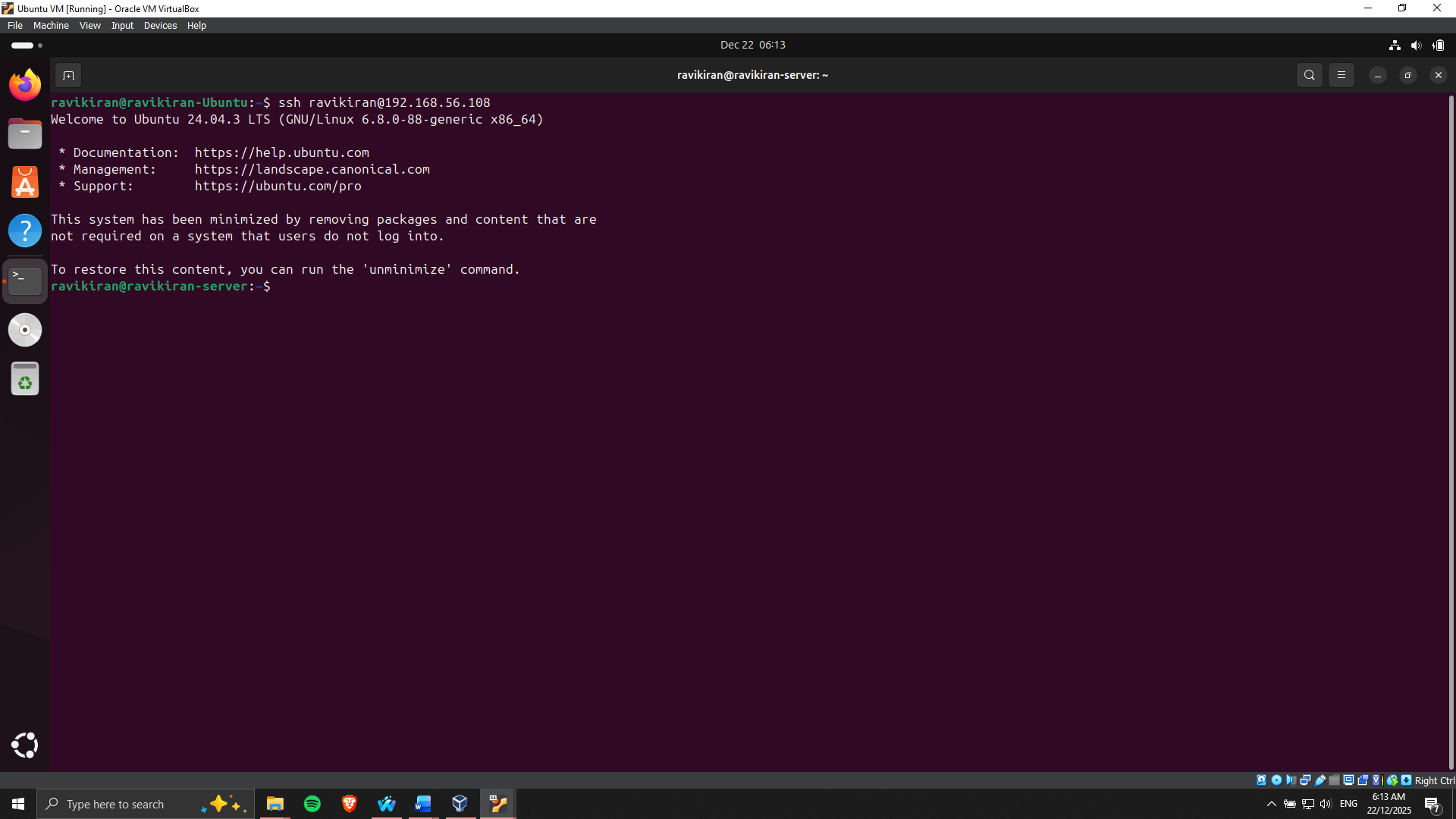


Figure 9: SSH to the server successful

# **Phase 5: Advanced Security and Monitoring Infrastructure**

**Access Control Implementation**

A fundamental security mechanism that was installed was Mandatory Access Control (MAC) with the use of AppArmor. AppArmor profiles of the critical applications were set up such that the resources and files that they could access are restricted. The threat of privilege escalation and unauthorized access was reduced by having strict access rules [7]. The sudo aa-status command examined the AppArmor status, which confirmed that all the required profiles were in action (Figure 10). Each step was documented in order to trace enforcement on the profile and report violations to ensure transparency and maintainability.

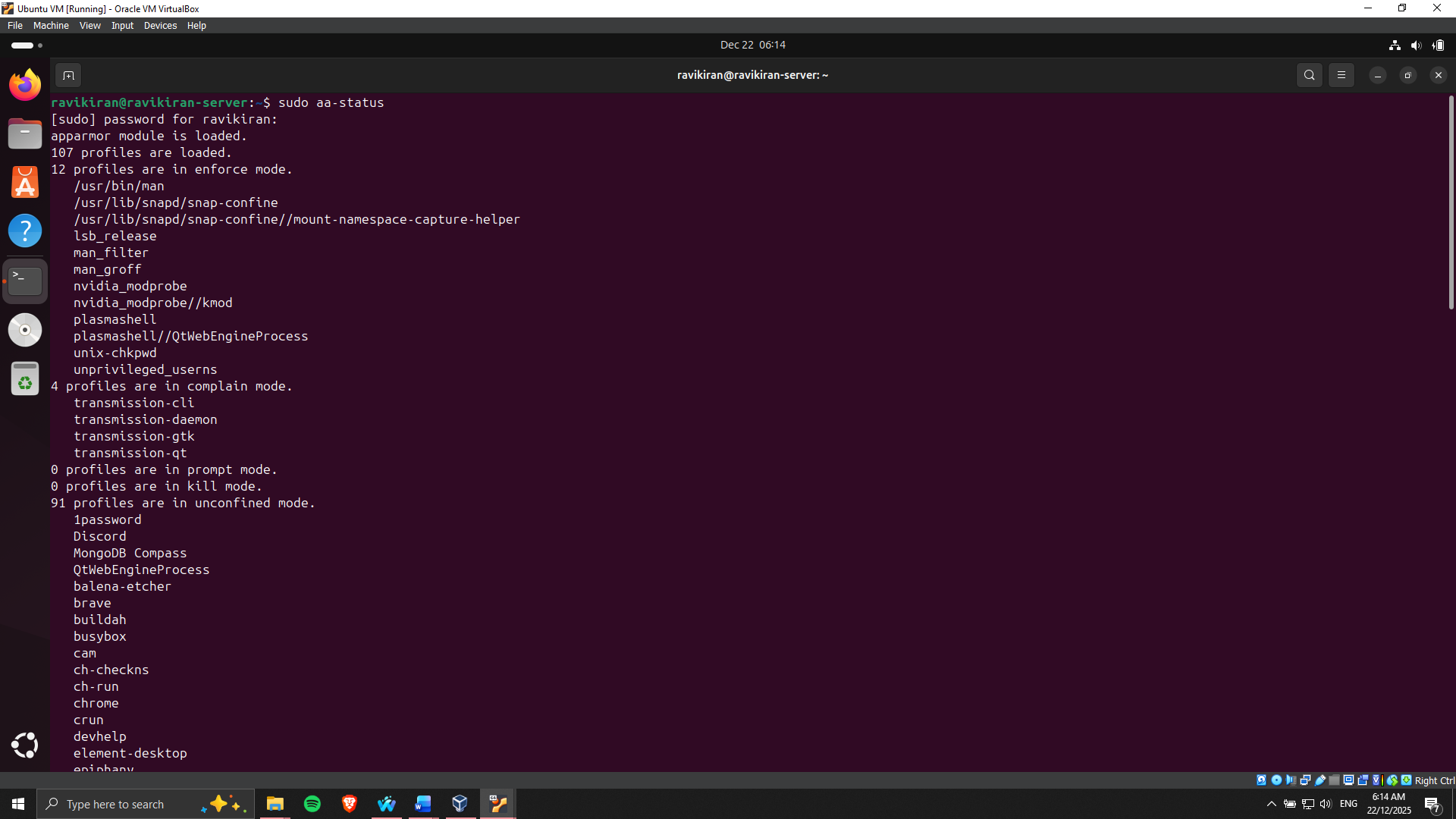


Figure 10: AppArmor Status verified

**Automatic Security Updates**

In order to perform continuous safeguarding against vulnerabilities, unattended upgrades were established. This helped to guarantee that security patches were crucial and automatically updated and installed. A configuration file apt.conf.d/50unattended-upgrades in the /etc was changed to provide all the security repositories. The implementation verification was also accomplished, proving the fact that updates were planned and implemented properly (Figure 11).



Figure 11: Unattended Upgrades enabled

Intrusion Prevention with Fail2ban

To have better intrusion detection, Fail2ban was installed and configured to check on SSH login attempts. Custom settings were used to block an IP address upon a specified number of failed attempts to log in. This minimally decreases the chances of brute force attacks [8]. The active status of fail2ban was verified with sudo systemctl status fail2ban and indicated that it worked correctly (Figures 12 and 13). Logs were monitored to confirm the work of banned IPs and also to confirm that the service was working as desired.



Figure 12: Fail2ban configured for SSH

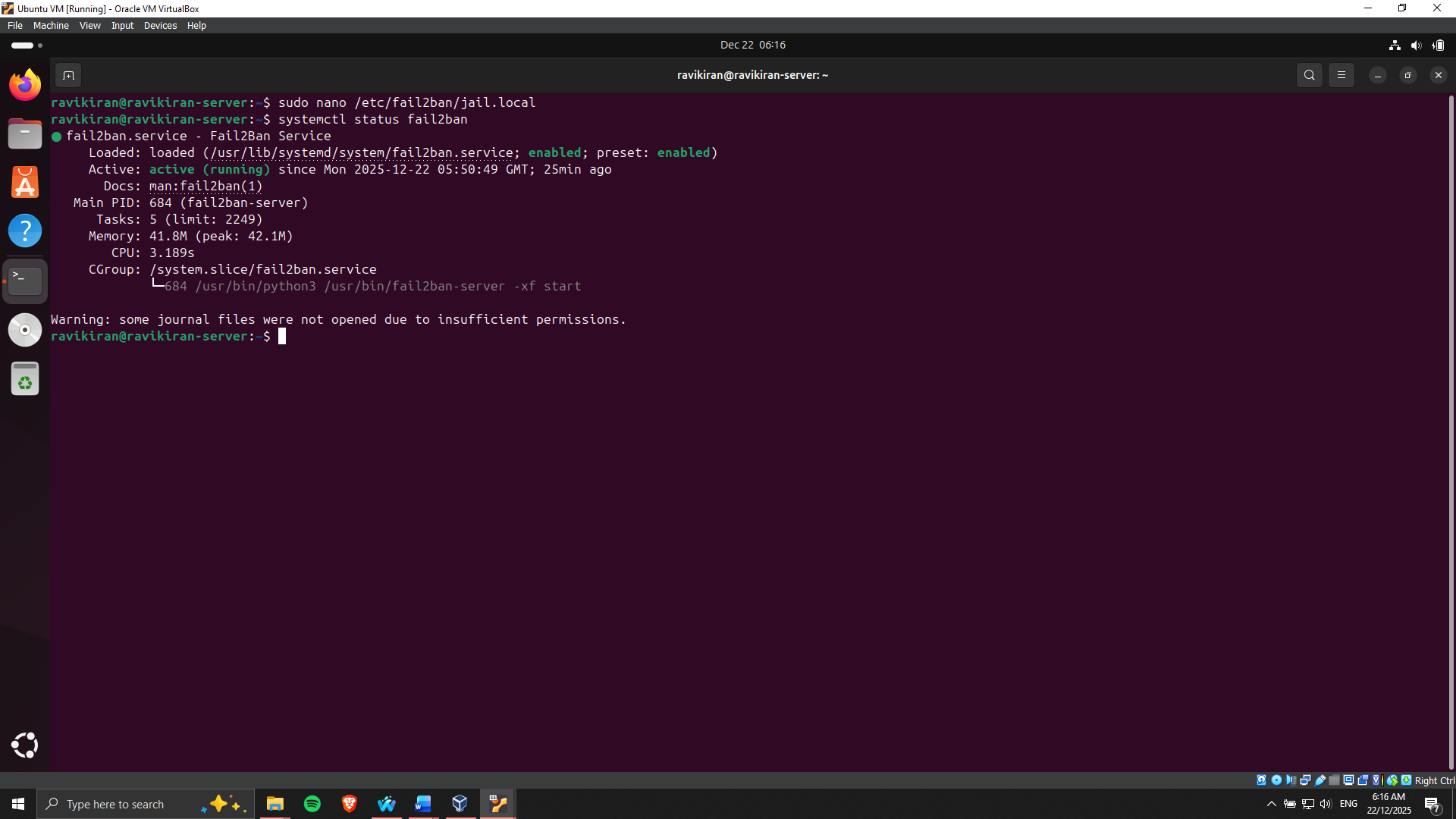


Figure 13: Fail2ban running on Server

**Security Baseline and Monitoring Scripts**

Security baseline verification script(security-baseline.sh) was developed to enable the automation of all security configurations when modifying Phases 4 and 5. The script analyses SSH configurations, firewall policies, user rights, AppArmor policies, Fail2ban configurations and automatic updates and leaves a report summary (Figures 14 and 15).

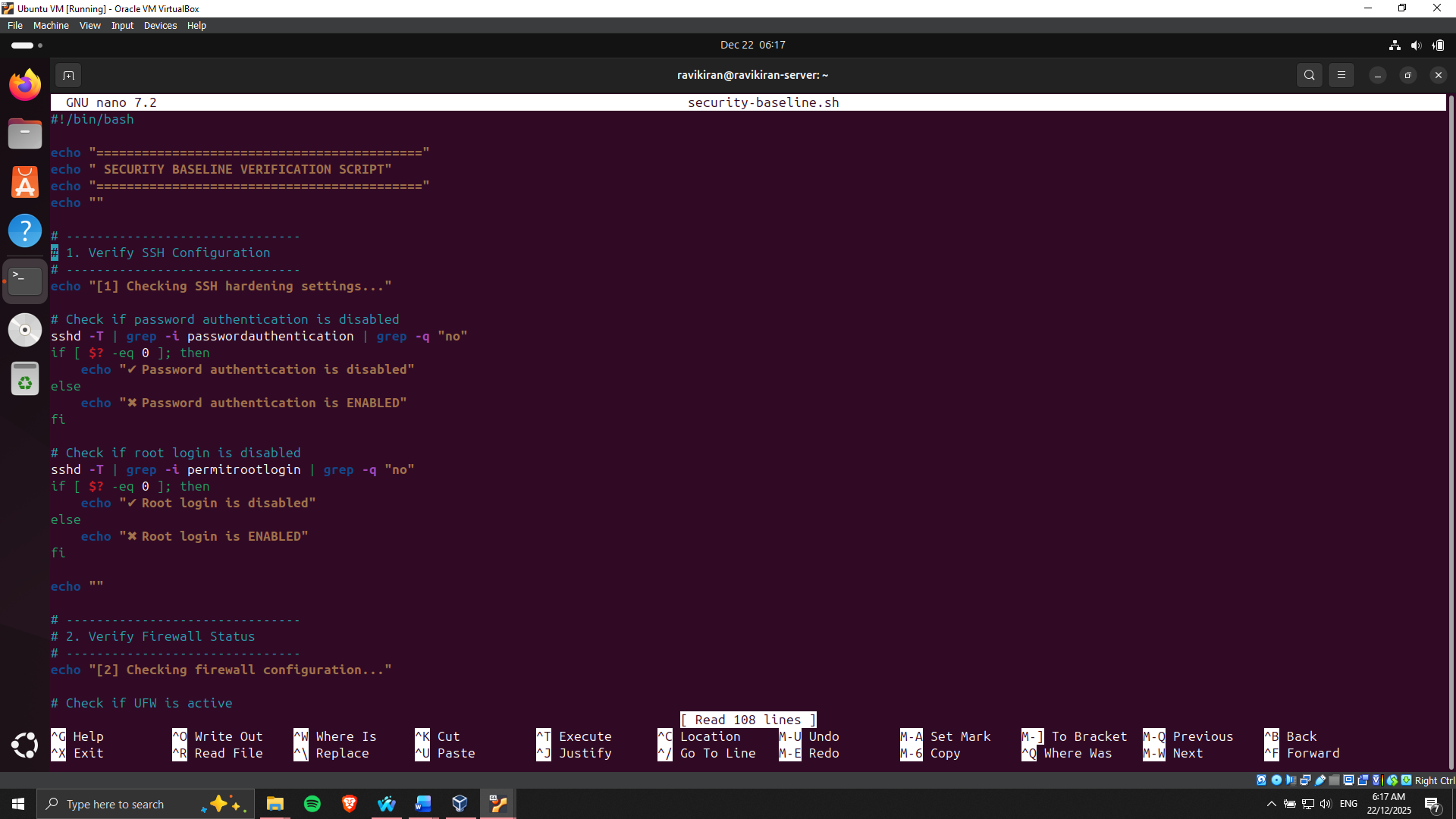


Figure 14: Code for capturing security baseline

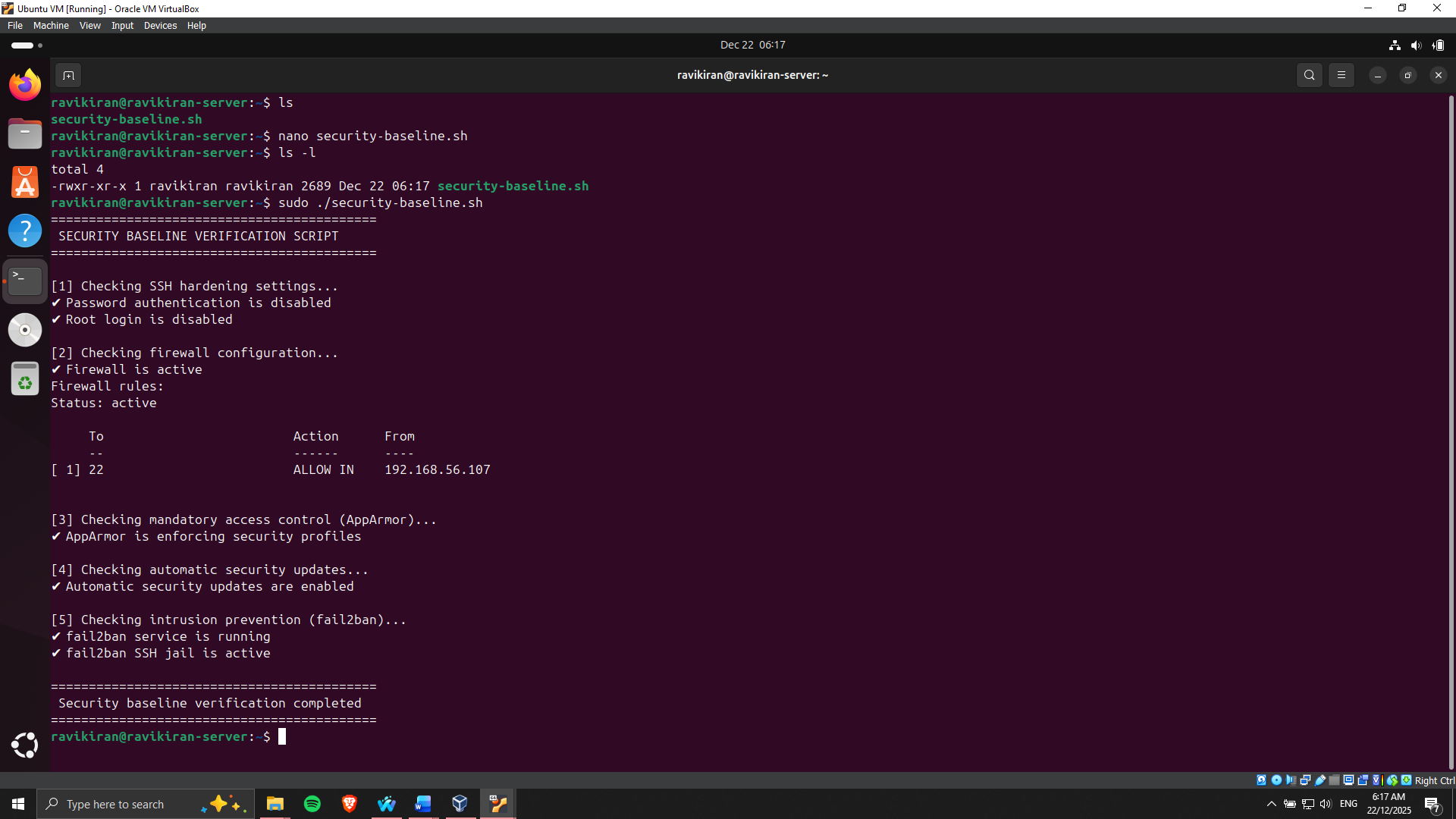


Figure 15: Security baseline captured

There was also the creation of a remote monitoring script (monitor-server.sh), which can be run out of the workstation. This script links to the server using SSH and obtains performance data of CPU, memory, disk usage, and network usage. This enables monitoring in real time and collection of past data, which can be used to analyse (Figures 16 and 17).

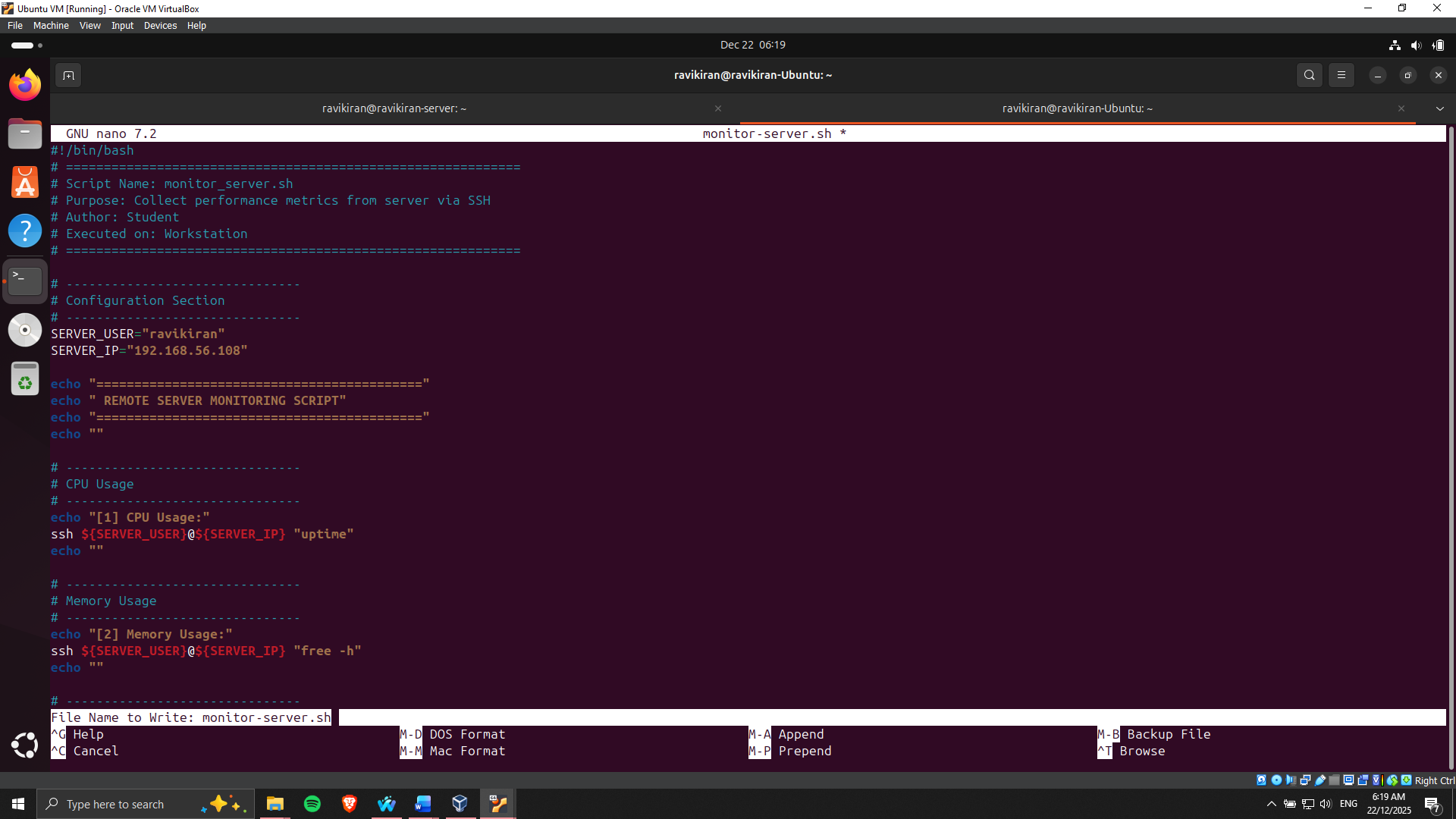


Figure 16: Script for monitoring server

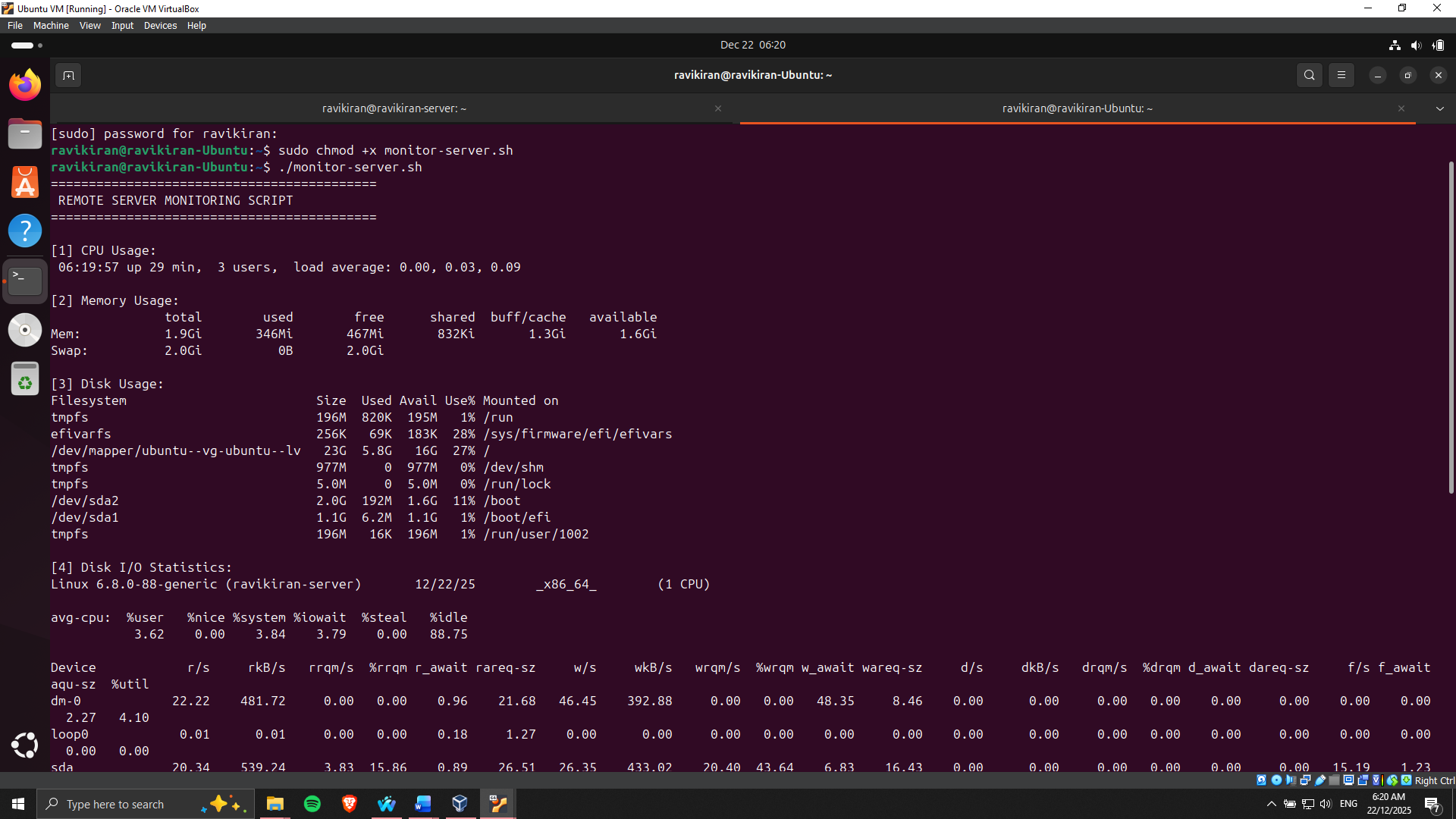


Figure 17: Monitoring of the server completed

In total, Phase 5 helped to improve the role of the server and set the framework for effective monitoring. Such assurances are both protection measures against threats and the possibility of testing the performance of the server under increased or decreased conditions.

# **Phase 6: Performance Evaluation and Analysis**

**Testing Methodology**

Remote testing performance testing was done through remote SSH with scripts and monitoring tools. Measures of CPU usage, memory consumption, disk I/O, network throughput, system latency and service response times were measured against each of the applications chosen. Measles under idle conditions was taken to form a baseline [9]. Applications-specific stress tests were then used to load test and test the behavior of the system as if it were under a heavy load.

**Workload Simulation**

The selection of applications used to represent different resource-heavy workloads, namely CPU-intensive applications, RAM-intensive applications, I/O-intensive applications, network-intensive applications and server applications. Stress-ng was employed during CPU matter testing by simulating the high processor load, leading to the high peaks of the CPU usage (Figure 18).



Figure 18: High CPU Usage

Applications with high memory requirements, including database processes, were also simulated to keep track of the trends of RAM consumption and determine the extent of swap use (Figure 19). Network-intensive tests were done with iperf3 and ping measurements to check disk I/O via file read/write, and network-intensive tests with the same.

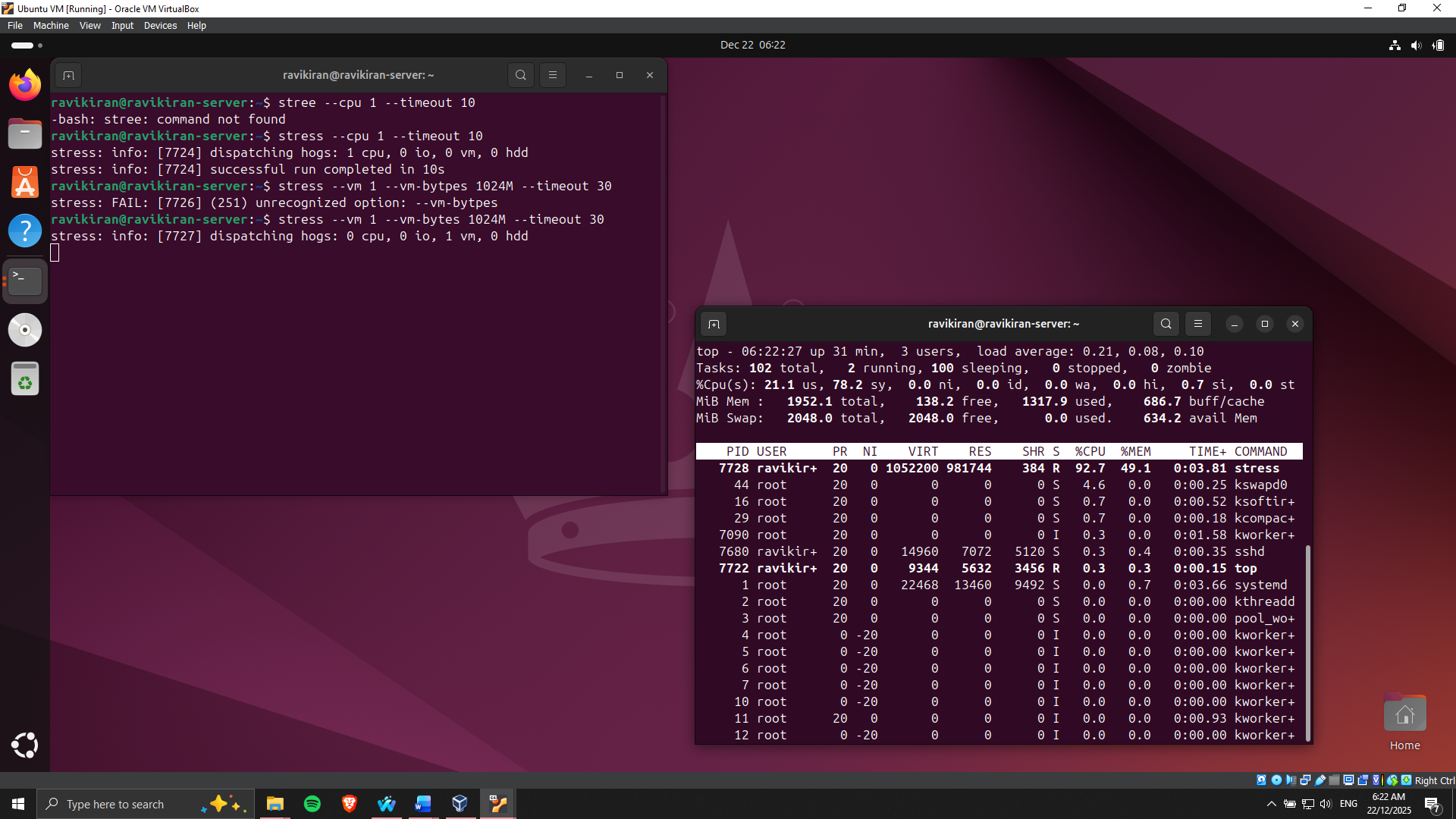


Figure 19: High RAM Usage simulated

**Performance Analysis**

The data that was obtained through monitoring tools and scripts was merged into organized tables in order to compare them. To visualise the trends of resource utilisation and determine bottlenecks in performance, graphs and charts were created. The first test that was made found that some of the applications had an ability to create memory contention and extreme CPU spikes that affected the service response times.

**Optimisation Measures**

Two major optimisations were made. The resource share was first changed with the help of cgroups to ensure that a single application did not hog resources away. Second, the I/O scheduling parameters were adjusted in a way that maximizes the disk throughput of the I/O-intensive applications. Post-optimisation testing revealed that there were improvements that could be measured, such as a fall in latency, the CPU peaks were also reduced, and the use of the memory became more stable [10].

# **Phase 7: Security Audit and System Evaluation**

**Security Scanning using Lynis**.

Lynis, which is an open source security auditor, was used to scan the server. Lynis conducts a thorough scanning of system files, system configurations and services, giving valuable information about vulnerabilities and misconfigurations [11]. Checks that were performed during the audit were: outdated package, insecure file permissions and possible vulnerability of authentication mechanisms. The scanned result has verified that the security measures put in place earlier were working, and the possible factors for slight improvements (Figure 20).

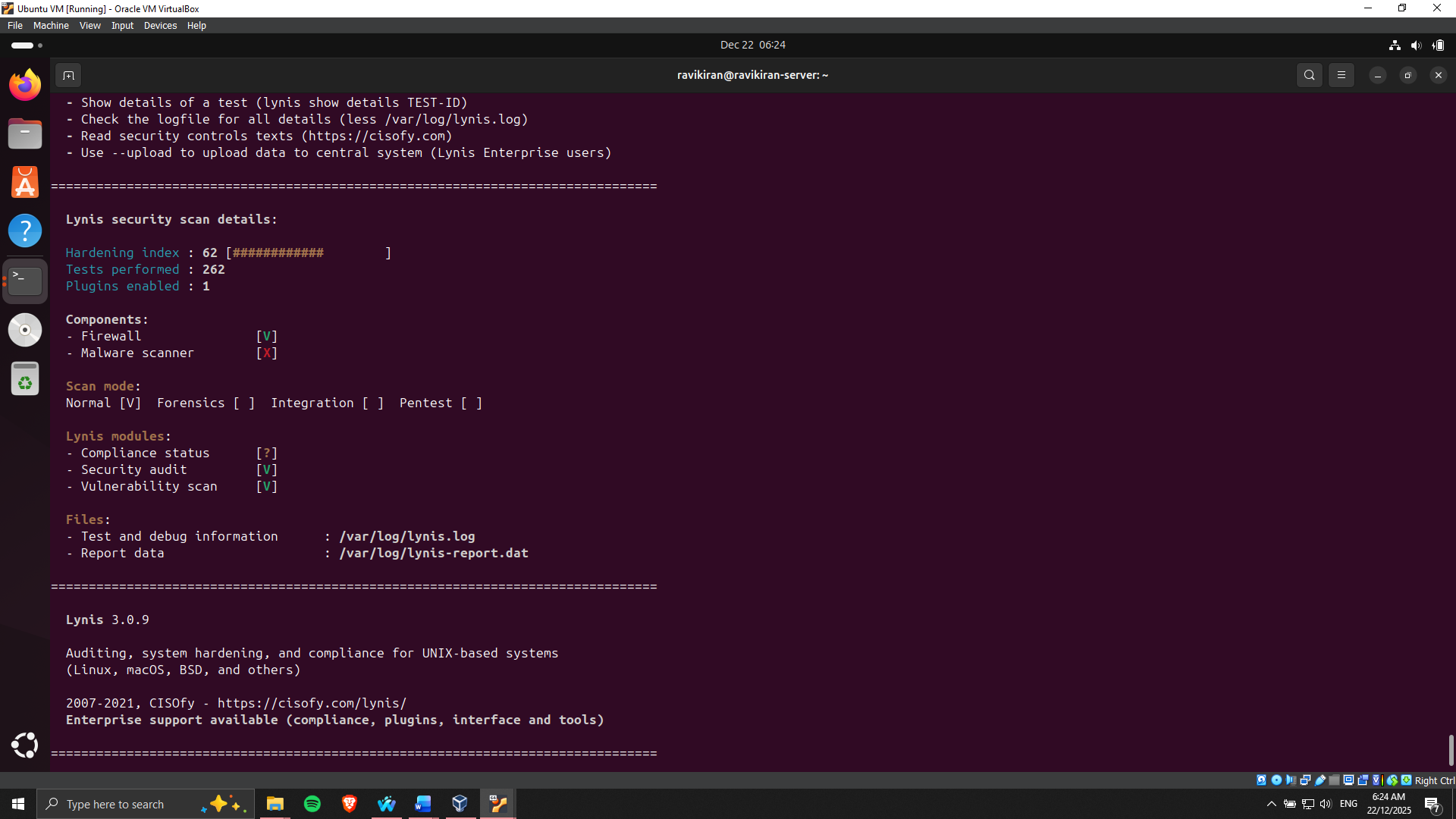


Figure 20: Lynis Audit completed

**Network Security Testing**

The evaluation of network security was performed with the help of Nmap, which analyses open ports and determines the services that are running and their versions. The scan made sure that only necessary services were open and all the unnecessary ports were closed [12]. The results provided by Nmap confirmed that access permissions were set according to the firewall configuration that only allowed SSH entry into the authorized workstation, but denied any other outside entry (Figure 21).

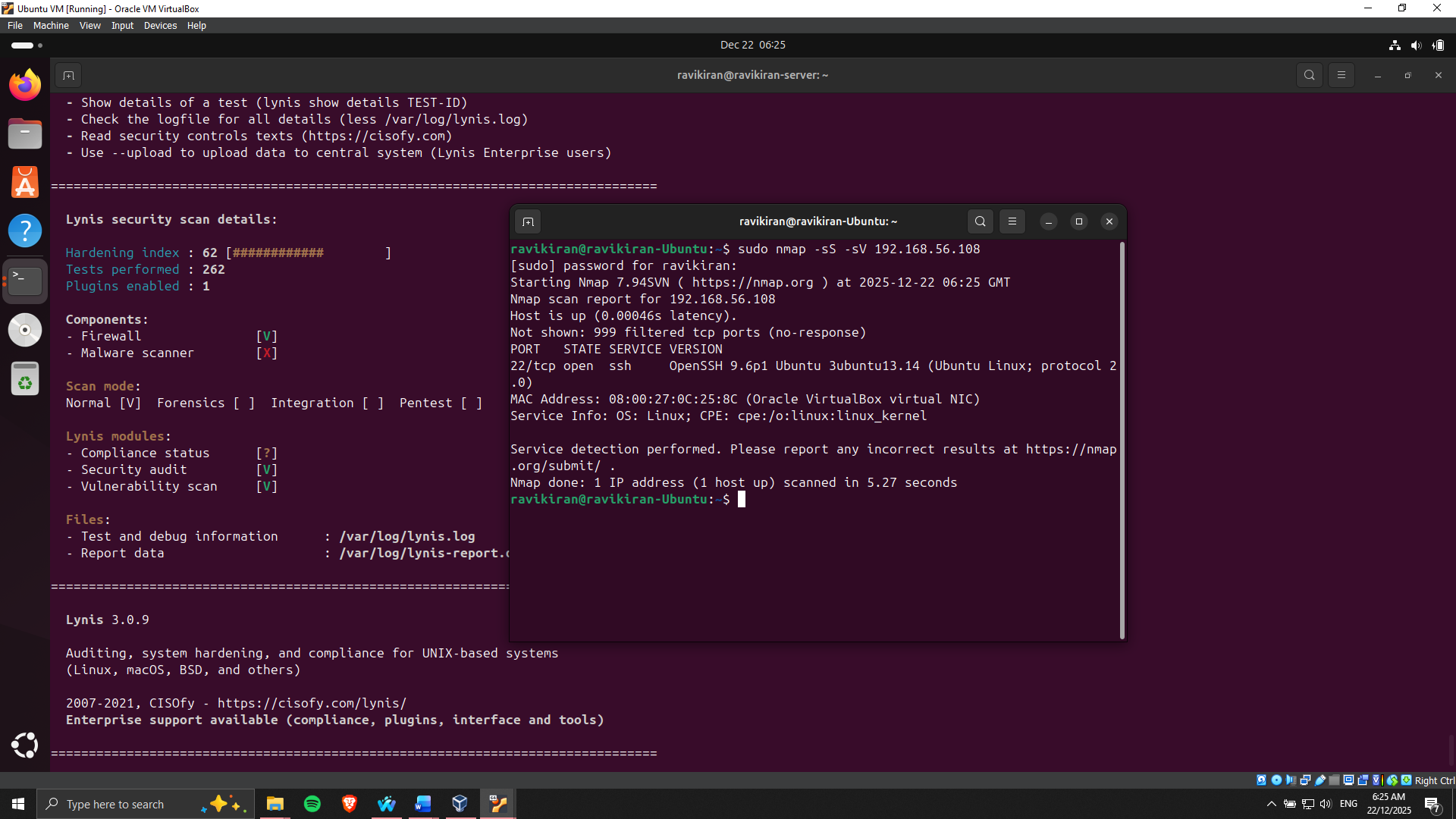


Figure 21: Nmap Scan results

**Service Audit and Access Control**.

Other access control checks involved AppArmor profile, user privileges and sudo access. All the profiles were on and imposing restrictions, and the user accounts were under the principle of least privilege. Service audit was used to check on running processes, and only necessary services were running, and unwanted daemons were shut down [13]. This will decrease the attack surface of the server and enhance the general security posture (Figure 22).

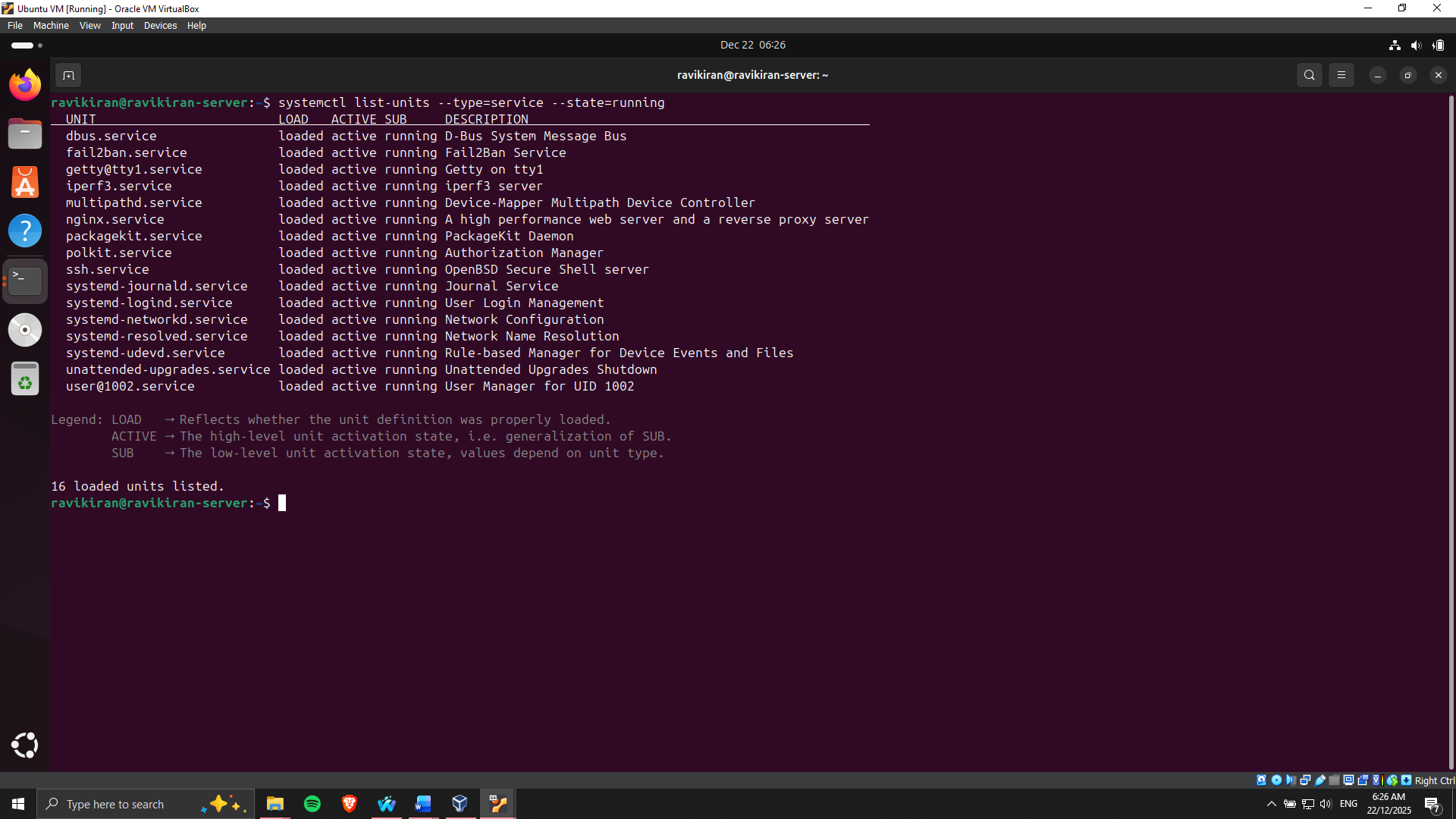


Figure 22: Services running on Linux Server

Review System configuration, System administration Paths

In the end, the SSH key-based authentication, automatic updates, Fail2ban, and other tools were reviewed to be properly implemented and activated. Best practices were aligned with configuration files, and all the monitoring and security baseline verification scripts were successfully executed.

# **Conclusion and Reflection**

This test presented an overall investigation on Linux server management, security applications, and server performance. By using the Ubuntu Server headless deployment and remote SSH administration, I have gained a pragmatic skill in configuring systems using Jason, firewall and user management, access control through AppArmor, and automated monitoring and security checks. The issue of resource management and optimisation was evidenced in performance testing of various loads and workloads, where it was necessary to guarantee the stable and efficient work of servers.

Being retrospective about the process, the progressive approach enhanced systematic problem-solving and critical thinking. The interaction between security, performance and system reliability was highlighted in the planning, implementation, and analysis stages. The experience has enhanced my knowledge of the professional practice of Linux administration and equipped me with the skills to put them into practice in a real-world setting: how to keep the operations as efficient as possible with strong security at the same time.

# **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] A. Ali, H. A. Maghawry, and N. Badr, “Performance testing as a service using cloud computing environment: A survey,” 2022, [Online]. Available: <https://www.researchgate.net/publication/362232172_Performance_testing_as_a_service_using_cloud_computing_environment_A_survey>

[3] Z. M. Jiang, A. E. Hassan, G. Hamann, and P. Flora, “Automated performance analysis of load tests,” in *2009 IEEE International Conference on Software Maintenance*, IEEE, Sep. 2009, pp. 125–134. <https://www.researchgate.net/publication/221307754_Automated_Performance_Analysis_of_Load_Tests>

[4] I. Red Hat, “Red Hat® Enterprise Linux® Security Target,” pp. 1–37, 2023, [Online]. Available: <https://www.commoncriteriaportal.org/nfs/ccpfiles/files/epfiles/st_vid11309-st.pdf>

[5] G. Prajapati and N. Khare, “A Comparative Study of Software Firewall on Windows and Linux Platform,” *Int. J. Comput. Technol.*, vol. 14, no. 8, pp. 5967–5978, Jun. 2015, <https://www.researchgate.net/publication/324985984_A_Comparative_Study_of_Software_Firewall_on_Windows_and_Linux_Platform>

[6] H. S. Hawedi, O. A. Bentaher, and K. E. I. Abodhir, “REMOTE ACCESS TO A ROUTER SECURELY USING SSH,” *J. Acad. Forum*, vol. 5, no. 1, pp. 174–189, Jan. 2021, <https://www.researchgate.net/publication/350567767_REMOTE_ACCESS_TO_A_ROUTER_SECURELY_USING_SSH>

[7] M. Alviano and P. Sestito, “User Armor: An Extension for AppArmor,” 2025, [Online]. Available: <https://www.researchgate.net/publication/390136257_User_Armor_An_Extension_for_AppArmor>

[8] C. Lai, A. Chavez, C. B. Jones, and N. Jacobs, “Review of Intrusion Detection Methods and Tools for Distributed Energy Resources,” 2021, [Online]. Available: [https://www.researchgate.net/publication/349699828\_Review\_of\_Intrusion\_Detection\_Methods\_and\_Tools\_for\_Distributed\_Energy\_Resources](https://www.researchgate.net/publication/349699828_Review_of_Intrusion_Detection_Methods_and_Tools_for_Distributed_Energy_Resources%20)

[9] J. Noor, M. G. Hossain, M. A. Alam, A. Uddin, S. Chellappan, and A. B. M. A. Al Islam, “svLoad: An Automated Test-Driven Architecture for Load Testing in Cloud Systems,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, IEEE, Dec. 2018, pp. 1–7. [https://www.researchgate.net/publication/331270058\_svLoad\_An\_Automated\_Test-Driven\_Architecture\_for\_Load\_Testing\_in\_Cloud\_Systems](%20https:/www.researchgate.net/publication/331270058_svLoad_An_Automated_Test-Driven_Architecture_for_Load_Testing_in_Cloud_Systems)

[10] S. Pargaonkar, “A Comprehensive Review of Performance Testing Methodologies and Best Practices: Software Quality Engineering,” *Int. J. Sci. Res.*, vol. 12, no. 8, pp. 2008–2014, Aug. 2023, <https://www.researchgate.net/publication/375450774_A_Comprehensive_Review_of_Performance_Testing_Methodologies_and_Best_Practices_Software_Quality_Engineering>

[11] A. Romanovych and M. .K, “Integrating Compliance Scanners with Custom UNIX Scripting,” 2020, [Online]. Available: [https://www.researchgate.net/publication/394306071\_Integrating\_Compliance\_Scanners\_with\_Custom\_UNIX\_Scripting](https://www.researchgate.net/publication/394306071_Integrating_Compliance_Scanners_with_Custom_UNIX_Scripting%20)

[12] S. Liao *et al.*, “A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments,” in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, Oct. 2020, pp. 64–71. <https://www.researchgate.net/publication/348826485_A_Comprehensive_Detection_Approach_of_Nmap_Principles_Rules_and_Experiments>

[13] A. Kern and R. Anderl, “Using RBAC to enforce the principle of least privilege in industrial remote maintenance sessions,” *2018 5th Int. Conf. Internet Things Syst. Manag. Secur. IoTSMS 2018*, pp. 107–114, 2018, <https://sweet.ua.pt/jpbarraca/course/iaa/papers/Using%20RBAC%20to%20Enforce%20the%20Principle%20of%20Least%20Privilege%20in%20Industrial%20Remote%20Maintenance%20Sessions.pdf>