Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2025), B.Sc. in CSE (Day)

SentinelShield: Advanced Linux Security Hardening & Threat Audit Framework

Course Title: Operating System Lab
Course Code: CSE-402
Section: 231-D2

Students Details

Name	ID
Promod Chandra Das	231002005
Chinmoy Debnath	231902029

Submission Date: 02.12.25
Course Teacher's Name: Md. Shoab Alam

[For teachers use only: Don't write anything inside this box]

Lab Project Status

Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

SentinelShield is a Bash-based Linux security auditing and hardening framework designed to automatically detect vulnerabilities and strengthen system security. It scans the system for weak configurations, insecure permissions, exposed ports, unnecessary services, and potential attack vectors. Using a policy-driven approach, it evaluates the system against predefined security standards, applies corrective hardening actions, and generates detailed security reports. SentinelShield also maintains backups and supports rollback to ensure safe and reliable system modifications.

1.2 Motivation

Linux systems are widely used in servers, cloud platforms, and enterprise infrastructures. However, most security breaches occur due to simple misconfigurations, weak SSH settings, and poorly managed permissions. Manual security auditing is slow, inconsistent, and prone to human error. Many organizations lack lightweight and customizable tools that can automatically assess and secure Linux environments.SentinelShield is motivated by the need for a simple, fast, and automated framework that ensures consistent system security and reduces administrative workload.

1.3 Problem Definition

Modern Linux environments contain numerous security risks arising from improper configurations and overlooked vulnerabilities. Issues such as world-writable files, open ports, SUID/SGID binaries, insecure SSH settings, and unnecessary services often remain undetected. Administrators struggle to manually maintain security across multiple systems due to complexity and time constraints.

1.3.1 Problem Statement

Complex Engineering Problem:

"To develop an automated, policy-driven Linux security framework capable of analyzing the system's entire attack surface, detecting multi-layered vulnerabilities, applying safe and reversible hardening actions, and ensuring consistent operation across diverse Linux distributions using Bash scripting."

This is a complex engineering problem because it requires:

- Cross-domain system analysis (filesystems, services, permissions, networking)
- Safe automated configuration editing
- Policy interpretation and validation logic
- Backup, rollback, and logging mechanisms
- Ensuring compatibility across heterogeneous environments

1.4 Design Goals/Objectives

The primary objectives of SentinelShield are:

1. **Automated Vulnerability Scanning:**

   Detect insecure permissions, misconfigured services, SSH weaknesses, and exposed attack surfaces.

2. **Policy-Based Validation:**

   Use a customizable INI-style security policy to define standards for system configuration.

3. **Automated System Hardening:**

   Apply safe fixes by adjusting permissions, modifying SSH settings, enabling unnecessary services, and securing system resources.

4. **Backup and Rollback Features:**

   Store backups of all modified configurations and support reverting to previous states.

5. **Detailed Security Reporting:**

   Generate structured reports including identified risks, actions taken, and an overall security score.

6. **Lightweight & Portable Implementation:**

   Build the entire system using pure Bash to ensure compatibility and low resource usage.

Table 1.1: Summary of the attributes touched by the mentioned project

Name of the P Attributes	**Explain how to address**
P1: Depth of knowledge required	The project requires strong understanding of Linux internals, file permissions, process management, SSH security, network services, and Bash scripting. It demands advanced knowledge of cybersecurity concepts and OS-level hardening mechanisms.
P2: Range of conflicting requirements	The tool must secure the system without disrupting essential services, balance automation with user safety, maintain compatibility across distributions, and enforce strict security while preserving system usability.
P3: Depth of analysis required	The project performs multi-layer analysis of permissions, services, ports, cron jobs, configurations, and SSH rules. It evaluates each item against policy standards, requiring deep technical inspection and logical decision-making.
P4: Familiarity of issues	The project deals with real-world security issues such as misconfigured SSH, world-writable files, insecure services, privilege escalation paths, and unnecessary running daemons that commonly exist in Linux systems.
P5: Extent of applicable codes	The project applies Linux hardening best practices, Bash automation techniques, security benchmarks (CIS-like standards), permission models, and safe configuration editing methods with backup and rollback mechanisms.
P6: Extent of stakeholder involvement and conflicting requirements	Different stakeholders (administrators, DevOps teams, security officers, and end-users) have varying needs such as maximum security, high uptime, ease of maintenance, and system stability. The project balances these through customizable security policies.
P7: Interdependence	The system interacts with multiple Linux subsystems such as filesystems, SSH, services, processes, and network configurations. Each component is interconnected, requiring coordinated scanning, hardening, and safe automation to avoid system conflicts.

1.5 Application

SentinelShield can be applied in multiple practical contexts:

- **Enterprise Server Security:**
  Automatically harden production Linux servers and reduce attack risks.
- **DevOps & System Administration:**
  Integrate with deployment pipelines to enforce security standards during server setup.
- **Cybersecurity Training & Education:**
  Provide learners with a real-world tool to understand Linux security mechanisms.
- **Cloud Infrastructure Protection:**
  Secure virtual machines in cloud environments like AWS, Azure, and GCP.

Personal Security Use:

Strengthen personal Linux machines to protect against unauthorized access.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Start the section with a general discussion of the project [1] [2] [3].

2.2 Project Details

In this section, you will elaborate on all the details of your project, using subsections if necessary.

2.2.1 Subsection_name

Figure 2.1: Figure name

You can fix the height, width, position, etc., of the figure accordingly.

2.3 Implementation

All the implementation details of your project should be included in this section, along with many subsections.

2.3.1 Subsection_name

This is just a sample subsection. Subsections should be written in detail. Subsections may include the following, in addition to others from your own project.

The workflow

Tools and libraries

Implementation details (with screenshots and programming codes)

Each subsection may also include subsubsections.

2.4 Algorithms

The algorithms and the programming codes in detail should be included .
Pseudo-codes are also encouraged very much to be included in this chapter for your project.

- Bullet points can also be included anywhere in this project report.

Algorithm 1: Sample Algorithm

```
   Input: Your Input
   Output: Your output
   Data: Testing set $x$
1  $\sum_{i=1}^{\infty} := 0$                              // this is a comment
   /* Now this is an if...else conditional loop              */
2  if Condition 1 then
3      Do something                                  // this is another comment
4      if sub-Condition then
5          Do a lot
6  else if Condition 2 then
7      Do Otherwise
       /* Now this is a for loop                             */
8      for sequence do
9          loop instructions
10 else
11     Do the rest
   /* Now this is a While loop                               */
12 while Condition do
13     Do something
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Discuss the experimental setup and environment installation needed for the simulation of your outcomes.

3.1.1 Subsection

3.1.2 Subsection

3.2 Results Analysis/Testing

Discussion about your various results should be included in this chapter in detail.

3.2.1 Result_portion_1

The results of any specific part of your project can be included using subsections.

3.2.2 Result_portion_2

Each result must include screenshots from your project. In addition to screenshots, graphs should be added accordingly to your project.

3.2.3 Result_portion_3

Each result must have a single paragraph describing your result screenshots or graphs or others. This is a simple discussion of that particular portion/part of your result.

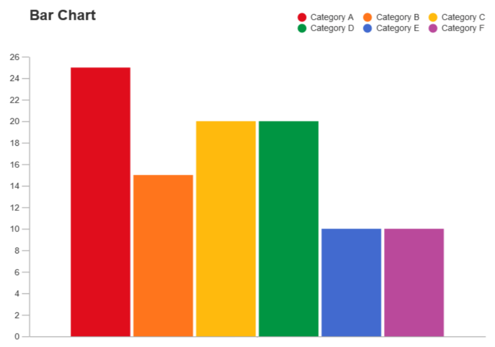

Figure 3.1: A graphical result of your project

3.3 Results Overall Discussion

A general discussion about how your result has arrived should be included in this chapter. Where the problems detected from your results should be included as well.

3.3.1 Complex Engineering Problem Discussion

[OPTIONAL] In this subsection, if you want, you can discuss in details the attributes that have been touched by your project problem in details. This has already been mentioned in the Table 1.1.

Chapter 4

Conclusion

4.1 Discussion

Discuss the contents of this chapter and summarized the description of the work and the results and observation. Generally, it should be in one paragraph.

4.2 Limitations

Discuss the limitations of the project. Limitations must be discussed, with the help of some critical analysis.

4.3 Scope of Future Work

Discuss the future work of the project, that is your plans for more work and extension of your project.

References

[1] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017.

[2] Douglas Laney. 3d data management: controlling data volume, velocity and variety. gartner, 2001.

[3] MS Windows NT kernel description. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm. Accessed Date: 2010-09-30.