

## **Abstract**

Against the backdrop of increasing cyber threats, firewall configuration has emerged as an essential element of organisational cybersecurity. Specifically, under the UK's Cyber Essentials (CE) scheme, correct firewall configuration is a central control, a necessity for SMEs that must achieve minimum security standards. Yet, today self-assessment techniques are predominantly manual, time-demanding, and reliant on technical skills, creating an onerous burden for small and medium-size organisations that do not have specialist cybersecurity specialists. This research overcomes this shortfall by creating an automatic, dual-platform Firewall Cyber Essentials Self-Assessment tool for use by non-technologists in SMEs.

The tool also enables uploading of firewall rule sets that have been exported from Windows and Linux platforms and includes automated compliance verification based on CE firewall. With the help of open-source libraries and Python, the tool parses rule files, checks for key misconfigurations like open RDP/SSH ports, permissive ANY-ANY rules, and the lack of default deny policies, and based on that, generates PDF reports with explicit compliance and remediation recommendations. The tool includes a graphical user interface based on Tkinter for ease of use, as well as log functionalities for traceability. It includes real time monitoring function .

Both secondary research—examining CE guidelines and available tools—and primary research—tool design, building, and validation with valid and invalid rule sets—was included in the methodology. Accuracy, usability, and effectiveness of the tool were tested with test cases based on actual configuration conditions. Tests indicated that the tool effectively identified CE-relevant misconfigurations and offered correct remediation guidance. The work concludes that an automated, lightweight compliance checker such as FCESA has the ability to greatly enhance firewall rule validation and CE alignment for SMEs, reducing security audit cost and complexity. The tool could be further improved by adding real-time monitoring and increased CE control coverage in subsequent work. Not only does the work provide a working artefact but also illustrates investigative sophistication and practical relevance, meeting academic and business cybersecurity objectives.

## **1. Introduction**

In the expanding digital economy, the cybersecurity environment keeps increasing in sophistication and threat level. Small and medium businesses (SMEs) end up disproportionately exposed due to limited technical and financial resources. The UK Government, through its National Cyber Security Centre (NCSC), has pointed out the necessity of core controls through the Cyber Essentials (CE) scheme. Of the five key CE controls, firewall configuration emerges as one of the most vital aspects in defining a sound network perimeter. Yet, making firewall rules

CE compliant usually involves a laborious and skill-dependent process, posing challenges for SMEs with minimal or no cybersecurity infrastructural backup.

The work falls within the context of enhancing compliance for these types of organizations by making the assessment process automatic. The dual-platform Firewall Cyber Essentials Self-Assessment (FCESA) tool—a proposed solution—assists with simplifying compliance by using automated rule evaluation and intuitive report generation. Built with Python and running on Windows and Linux platforms, the tool takes exported firewall rules and compares them with CE-defined standards. The tool presents its findings in a normalized PDF format with qualitative feedback, enabling non-technical users to self-check for and correct their own misunderstandings without requiring technical intervention.

### **Problem Overview:**

Firewalls serve as the first line of cybersecurity defense, functioning as gatekeepers that manage incoming and outgoing network traffic. However, incorrect firewall rule configurations like open Remote Desktop Protocol (RDP) or Secure Shell (SSH) ports, default policy omissions, and permissive rules for any-to-any communications will disable the most robustly designed firewalls. SME environments frequently have these incorrectly configured environments due to the unavailability of specialized experts and the utilization of old or manually updated rule sets. Commercial offerings like Splunk, Windows Defender, or paid SIEMs that include firewall audit functionalities are too expensive or beyond the scope of SME environments.

### **Current Issues:**

Not only are manual CE self-assessments and manual audits time- and resource-consuming but they are also prone to human error. The majority of SMEs lack the skills necessary for the interpretation of sophisticated firewall schemes or accurately correlating them with CE standards. Neither do available solutions have automation, accessibility, or affordability. Even among freely available tools, there is excessive learning involved, minimal documentation, or weak interoperability with the CE control framework. This initiative proposes narrowing this gap by providing a stand-alone tool that SMEs will be able to utilize with minimal technical intervention.

### **Project Description:**

The FCESA tool has been written as a cross-platform, GUI-enabled tool using Python for ease of usage. It takes firewall rule sets that have already been exported from Windows (JSON) and Linux (plain-text) and employs built-in parsers that automatically identify most common CE violations. It runs locally and does not upload any sensitive information to an external server, maintaining confidentiality of the data. The tool output includes a PDF report that outlines rule by rule compliance problems and suggests remediations. It has also included logging for traceability of all the interactions. It reduces the technical entry barrier for SMEs that do not employ full-time IT staff.

### **Aims and Objectives:**

- Automating the firewall configuration self-assessment in accordance with Cyber Essentials standards.
- Supporting both Windows and Linux platforms using dual parser architecture.
- Helping SMEs discover and correct misconfigurations without requiring computer security expertise.
- Generating actionable, reportable compliance in PDF format.
- Evaluating the tool's accuracy, usability, and effectiveness using test scenarios.

### **Research Question and Originality:**

Is there an automated, GUI-enabled tool that can identify Cyber Essentials firewall misconfigurations in Windows and Linux environments accurately and assist with remediation without the need for specialized cybersecurity skills?

The innovation of the FCESA tool resides in its dual-platform compatibility, ease of use and accessibility for persons without technical skills, and total alignment with Cyber Essentials—an under-exploited but vital criterion for UK-serving organizations. All other available tools either fall into the category of excessive complication or have no straightforward mapping with CE. This work fills that niche with both scholarly value and functional application. Further, by correlating pulled rule content with CE controls and modeling real-world rule breaches, the tool provides a novel investigative methodology.

### **Feasibility, Commercial Context, and Risk:**

FCESA has feasibility as its design focus. It operates with lightweight technology such as Python and Tkinter, making it platform-independent and simple to deploy. External dependencies and cloud services are not engaged, thus decreasing both cost and complexity. From a business perspective, the tool has revenue value for MSPs and SME consultancy services that would like to include CE readiness assessments in their offerings. Potential risks include minimal testing with varied firewall environments, fluctuating CE guidelines, and the lack of real-time monitoring options—identified as possible future improvements. Ethically, the project has data minimization principles as its design focus, where there are no externally stored or transmitted sensitive logs or rule data.

### **Technological Significance and Overall Impact:**

From a technical standpoint, the project proposes a practical model for the evaluation of firewall rule efficacy by way of pattern identification and logic-based comparison with policy templates.

It is not merely a parser but an intelligence layer that interprets configuration in a context of compliance. The dual-platform support broadens its horizon of accessibility and pertinence, particularly in hybrid IT environments where Linux servers and Windows endpoints coexist. Furthermore, by presenting a GUI interface, the tool bridges a crucial gap in cybersecurity tools for SMEs—usability versus depth. This fits with contemporary cybersecurity projects aimed at democratizing security tools and simplifying the barrier of entry for compliance operations.

### **Report Structure:**

This report is divided into seven central chapters. The Literature Review reviews core CE needs, firewall configuration mistakes, and available compliance tools. The Methodology explains the design choices, software tools, and implementation plan. The Testing and Results chapter provides test cases and analysis of the results. The Evaluation and Conclusion critically considers project results, viability, and room for improvement. The report finishes with references and appendices such as screenshots, code snippets, and raw output. Through such an approach, the report fulfills the expectations of academic rigor while exercising practical relevance towards improving SMEs' cybersecurity stance.

## **2. Literature Review**

### **2.1 Introduction to the Field**

Cybersecurity has grown increasingly central as a focus for businesses around the world, with SMEs tending to be disproportionately affected. Firewall configuration forms one of the central pillars of cybersecurity within an organization and is most widely understood as the first in a series of defenses protecting from external threats. The UK's National Cyber Security Centre acknowledges this with its Cyber Essentials (CE) framework for compliance, with firewall configuration as one of its control areas. This review of the literature examines the principal cybersecurity tenets of firewall implementation, the importance of compliance schemes such as CE, and available tools and studies that drive the direction of this project.

Firewall configurations have traditionally been operated by hand or with enterprise tools that demand deep technical skills and investment. This disadvantages SMEs. NCSC and IASME literature suggests that SMEs regularly configure their firewalls incorrectly, leaving open key ports like RDP (3389) and SSH (22), and do not implement default deny policies. Incorrect configuration is associated with an increase in attack vectors for brute-force logins and remote code execution. There exists research that suggests automation and visualization tools will reduce the knowledge barrier and improve configuration reliability.

---

### **2.2 Cyber Essentials Framework and Its Importance**

Cyber Essentials is an industry-backed, government-supported scheme created for protecting organizations from typical cyber attacks. The five main control areas, according to NCSC technical guidance of 2022, are:

- Firewalls and Internet Gateways
- Secure configuration
- User access control
- Malware protection
- Patch management

CE dictates that organizations lock down access for services, institute default deny policies, and shun rules that provide broad, unmanaged access. Thus, the firewall configuration element most fits with perimeter defense mechanisms researched in conventional network security literature (Stallings, 2018).

It has been demonstrated through numerous studies, such as works presented by O'Hanlon (2021), that the majority of CE violations occur either due to misinterpreting the requirements or ineffective automated check mechanisms. For example, SMEs will typically permit inbound traffic on RDP or not limit outgoing traffic to untrusted networks. This implies that tools used for compliance should map their logic straight into CE expectations for effective feedback.

---

### **2.3 Significance of Automated Compliance Tools**

A study by Shah et al. (2020) highlights the importance of automation in upholding consistent and effective security protocols. Manual audit of the firewall runs the risk of human error and missed checks, particularly in those organizations with limited security specialists.

Compliance-as-code has caught on in DevOps and enterprise environments but, due to tool sophistication, is not yet widely applied in SMEs.

A key factor in the existing literature is that usability and efficacy are inversely related in cybersecurity tools. Splunk, for instance, while formidable, has a tremendous learning curve that necessitates higher-level configuration. Small businesses are less likely to stay with complicated solutions in return for simplicity, as demonstrated by research from Kim and Lin in 2019. This opens up the market for streamlined, user-friendly assessment tools with a focus on CE compliance.

---

## 2.4 Current Solutions and Drawbacks

There are some open-source and proprietary products that include firewall audit features. Products like Firewalld, GFW, and Iptables-save have rule inspection features but no automated CE rule validation. Splunk and other SIEM platforms have log analysis and firewall traffic monitoring features but are typically too expensive or too complicated for SMEs.

Windows Defender ATP includes strong firewall monitoring but only works with a tightly integrated Windows environment and does not perform as well with hybrid environments where there are Linux endpoints or servers. Defender doesn't include CE-specific remediation guidance.

Current research tools (e.g., NetSPARQL, NetAudit) concentrate on identifying firewall anomalies and conflicts but do not map firewall policies explicitly to CE policies. That limited focus on compliance makes them less useful for SMEs that have to achieve the formal certification norm. Also, most tools do not provide Windows and Linux platform support, leading to interoperability problems with heterogeneous operating system environments.

This effort addresses these gaps by providing a light-weight, dual-platform tool that converts firewall rules into CE-conformant compliance checks, creates PDF reports, and provides remediation recommendations.

---

## 2.5 Port-Specific Threats: RDP and SSH

Remote Desktop Protocol and Secure Shell are popular tools for remote administration but are abused by attackers. RDP ranks among the top three compromised services in ransomware attacks, as reported by McAfee's 2021 threat report. SSH open ports have also been utilized in brute-force and credential-stuffing attacks (IBM X-Force, 2022). CE requires such ports to be closed unless specifically necessary and secured by robust authentication measures.

While most firewall audit tools indicate open ports, they do not put these flags into context with CE needs. For instance, an open RDP port may be acceptable in a corporate environment but be a CE violation without two-factor authentication and IP whitelisting. Compliance assessment, therefore, should be standards-conscious and not merely technology reactive.

---

## 2.6 ANY-ANY Rules and Default Deny Policies

The application of broad ANY-ANY rules—where any source can access any destination on any port—is deemed a key misconfiguration. CE stipulates that only access for services or

applications that are explicitly needed for the business to function should be granted (NCSC, 2022). However, research (Simmons et al., 2020) indicates that ANY-ANY rules are prevalent, frequently created by old policies or incorrectly configured default templates.

Default-deny policies are the opposite of permissive rule sets. CE mandates the default action of a firewall to be denial unless expressly permitted. Many default setups of commercial firewalls or tools such as iptables and ufw are likely default-accept, particularly on older versions of Linux. The compliance tool must thus identify not only the occurrence of unsafe rules but also the lack of secure defaults.

---

## 2.7 Linux and Windows Rule Structures

An interesting factor in firewall evaluation is the structural difference between Linux and Windows rules. Linux rules, handled through iptables or UFW, tend to be stored in plain text, whereas Windows rules are accessed through PowerShell commands and tend to be written in JSON format for automation.

Huang and Chen (2020) highlighted that such distinctions render the creation of cross-platform compliance software more difficult. The majority of software addresses either environment, but not both. Dual-parser design used in this work draws on such research, with compatibility addressed by adapting rule parsing according to OS-specialized formats.

---

## 2.8 Literature-Driven Tool Requirements

Based on the reviewed research and standards, the most important requirements for the FCESA tool were determined as follows:

- **Dual-Platform Support:** Accommodating both Windows (PowerShell/JSON) and Linux
- **CE-Centric Evaluation:** Mapping logic to Cyber Essentials criteria
- **RDP/SSH Detection:** Detecting open remote ports and flagging them as non-compliant unless specifically justified
- **ANY-ANY and Default Policies:** Flagging wide-ranging rules and confirming existence of deny-all default policies

- **Automated Report Creation:** Offering PDF summaries and plain-language remediation recommendations for non-technical users
- **Usability for SMEs:** Reducing dependencies, providing GUI support, and keeping learning curves flat

These needs address not just technical but also usability demands highlighted in literature on SME cybersecurity preparedness.

---

## 2.9 Identification of Gaps

Several literature gaps informed this project's design:

- **Insufficient CE-Focused Tools:** All available tools are not aligned with CE needs, and SMEs have no way of interpreting the output.
- **No Dual-Platform Support:** There are few tools with consistent experiences on Linux and Windows.
- **Usability Shortfalls:** The tools frequently emphasize technical substance over accessibility.
- **Lack of Remediation Guidance:** Even when misconfigurations are flagged, users receive no advice on resolution.

These gaps confirm the reason why a specifically designed tool that unites cybersecurity understanding with CE compliance implementation becomes necessary.

---

## 2.10 Conclusions and Recommendations from Literature

Some conclusions arise from the reviewed literature:

- CE compliance, especially in firewall configuration, is inadequately addressed by current tools.
- Automated tools must be standards-aware, and not merely syntax-driven.



- Ease of use and dual-platform compatibility are essential for SME uptake.
- Remediation recommendations optimize tool usability by informing nontechnical users.
- Not only should evaluation be focused on detection but also on reportable action.

### **Recommendations for the project:**

- Enforce CE-specific rule validation logic.
- Emphasize GUI accessibility in order to reduce the barrier of entry.
- Verify tool accuracy with real-world rule samples and misconfigurations.
- Provide platform independence through independent Linux and Windows parsing logic.

## **3. Methodology**

This section describes the systematic methodology used in designing, developing, and validating Firewall Cyber Essentials Self-Assessment (FCESA) tool. The methodology is based both upon investigative and practice research with a hybrid of formal software development methodologies and critical testing. The development process is guided by an iterative, agile-influenced development cycle to design, test, and evaluate tool features incrementally, including a recently incorporated real-time monitoring capability, and to align with Cyber Essentials (CE) guidelines for firewall compliance. The methodology is based upon both academic rigour as well as practical considerations for usability and impact upon SMEs.

### **3.1 Research and Requirements Gathering**

The first stage of methodology was to identify a set of the most important functional and non-functional requirements for a CE-compliant firewall audit tool. This was based upon secondary research, mainly through an examination of the formal Cyber Essentials guidelines, IASME publications, and supporting research studies into SME cybersecurity issues. Close attention was paid to the five CE controls, with specific emphasis upon the "Boundary Firewalls and Internet Gateways" category.

From this research, a number of essential compliance requirements were established:

- RDP and SSH ports (ports 3389 and 22, respectively) should not be open to incoming connections.
- Inbound traffic must be subject to a default deny all policy.

- No rule for firewalls should permit universal any-to-any communication.

Furthermore, usability limitations for non-technical users were considered, including a need for a graphical user interface (GUI), user-friendly workflows, and auto-reporting. As a result of this research, it was clear that a specific research question was needed: Can a dual-platform, automated firewall tool help SMEs meet CE requirements for firewalls without requiring expertise intervention?

### 3.2 Tool Architecture and Design

The FCESA tool is designed as a light, standalone Python 3.11 application with an architecture incorporating well-spaced components for compliance analysis, rule parsing, PDF report generation, real-time monitoring, and audit logging. The user interface is provided through Tkinter to maintain platform-native interface compatibility. The tool is meant to be used both on Windows and Linux platforms, accepting exported firewall rule files in JSON (Windows) or plain text (Linux) formats.

During design, rule parsers were abstracted into specific functions to facilitate extensibility and future integration of additional control elements. Particular care was taken to make the tool portable and safe—no access to the Internet is needed, and everything is processed locally.

### 3.3 Integrating Real-Time Monitoring

In response to project scope refocusing and aligning with formal assessment brief guidelines, a mechanism for monitoring in real-time was incorporated into the tool. It allows for periodical monitoring of firewall configurations against CE compliance without having to manually relaunch the tool. It performs two principal functions:

- **Background Automation:** Periodically, and with a default interval (e.g., once every 10 minutes), automatically checks the last selected rule file.
- **Proactive Reporting and Alerts:** Creates new logs and reports if non-compliant configurations are found during monitoring.

Python's threading and time libraries are employed by the monitoring logic to spawn a background task that is non-blocking, running outside of the GUI's main thread. With this, the GUI is kept responsive as compliance checks are executed in the background.

A toggle control was included in the GUI to make it simple to turn monitoring on and off. It provides users with flexibility to have constant compliance verification or total control. The output of each monitoring cycle is recorded and stored in timestamped PDF reports for audit.

Integration involved reworking both file-handling and main event loop logic to prevent race conditions, file access errors, and redundant reporting, and to provide for proper error handling with try-except constructs and notification to users for all actual violations through pop-up messages, provided that the GUI is running at that time.

### 3.4 Tools, Technologies, and Implementation Stack

These key tools and technologies were employed by the project:

- **Python 3.11** – First-choice programming language based on its ease of use, cross-platform compatibility, and extensive ecosystem.
- **Tkinter** – Python's built-in GUI library for building interfaces.
- **PowerShell** - For exporting Windows Firewall rules through the command:  
`Get-NetFirewallRule | Get-NetFirewallPortFilter | ConvertTo-Json > firewall_rules_windows.json`
- **Iptables** – For viewing Linux firewall rules with:  
`sudo iptables -S > firewall_rules_linux.txt`
- **ReportLab** – Used for creating organized PDF compliance reports.
- **Logging Module** – Python's built-in logging utilised to log all events into `firewall_compliance_tool.log`.
- **Threading** – Allows real-time background monitoring with no impact on GUI responsiveness.

Implementation was based on Test-Driven Development (TDD) with each new module—parsing, report generation, or monitoring—being tested against mock rule files to be sure of expected behavior.

### 3.5 Parsing and Evaluation of Rules

There were two parsers created that parsed Windows and Linux formats for firewall rules. The Windows parser parsed the exported JSON file to look for violations of CE, which include:

- `"RemotePort": 3389 or 22` and direction is inbound and action is allow.
- Missing default inbound deny policy.

- Rules with "RemoteAddress": Any and "LocalAddress": Any.

The parser for Linux reads through every line of plain-text iptables output, searching for patterns such as:

- ACCEPT on ports 22 and 3389.
- `-A INPUT -s 0.0.0.0/0 -j ACCEPT` (ANY-ANY rule).
- Lack of DROP policy within the INPUT chain.

Each rule is tested against a compliance dictionary and failed checks are tracked for inclusion in the resulting PDF report with remediation recommendations.

### 3.6 GUI and User Experience

The GUI was designed with usability as its priority, since the tool targeted non-technical SME users. The user is provided with an ability to:

- Choose an operating system.
- Load a given firewall rule file with Browse.
- Conduct a compliance audit.
- Export PDF reports.
- Export log files.
- Enable real-time monitoring.

Feedback is provided through message boxes stating compliance or non-compliance with the system, with full results provided via the resultant report. The GUI also performs error handling for malformed files and omitted fields, giving users clear messages and sustaining strong logging.

### 3.7 Logging and Traceability

A persistent logging mechanism was incorporated into the tool to improve auditability. All meaningful events—compliance checking, file selection, monitoring trigger, error, and report—are saved in a time-stamp fashion. The log file

(`firewall_compliance_tool.log`) may be exported via the GUI for record-keeping or compliance audit. This conforms to CE guidelines for maintaining audit trails.

Logs have several purposes:

- Assist users and developers to troubleshoot issues.
- Provide transparency with real-time monitoring cycles.
- Provide historical evidence of compliance checks.

### 3.8 Testing and Validation Strategy

There were four categories of testing:

- **Unit Testing** – Affirmed that every function (for instance, parsing logic, compliance dictionary, report creation) functioned individually with controlled inputs.
- **Integration Testing** – Confirmed whether modules interacted as a group, including user interface and file behavior.
- **Scenario-Based Testing** – Used actual-choice firewall setups with proven CE violations to model realistic SME environments.
- **Stress Testing (Monitoring)** – Tested for stability of real-time monitoring for prolonged runs (6–12 hours) and looked for memory leakage, repeated reports, or thread failure.

Representative output for all tests was documented, and this was included as appendices for this report. Testing also verified correctness for remediation messages and fail/pass classification.

### 3.9 Ethical and Legal Aspects

Even though there is no collection or forwarding of personal data, ethical issues have been addressed. The local-only design prevents any rules being passed to third parties through firewalls. The tool does not make any modification to the actual firewall configuration and runs in read-only mode. With this design, privacy and integrity of SME systems are safeguarded, and professional cybersecurity ethics as well as legal requirements are met.

Also, being open-source, the tool encourages openness and supports academic best practices around reproducibility and peer review. Its usage, though, is meant for self-evaluation and not for intrusive audit purposes or enforcement measures.

### 3.10 Limitations and Risk Management

Although the tool reaches its main targets, there are certain limitations:

- It presumes that users are exporting the rules of a firewall properly.
- It currently does not handle advanced rule sets such as those implemented by UFW or nftables.
- Monitoring frequency is predetermined and won't necessarily be ideal for all environments.

These risks are addressed with user documentation, default protection (i.e., timestamping overwritten reports), and inherent error management. Future updates may include integration with scheduled tasks or improved parsing for third-party firewalls.

Ease of navigation, minimal inputs, and informative error handling were key design considerations. Usability tests, although initial, indicated confusion over file formats and the absence of confirmation messages, which led to iterative refinement of the GUI.

While GUI aspects are usually overlooked in technical projects, in this instance, they were key to the objective of supporting SMEs. Without an interface, the tool would have to be used from the command prompt—contradicting its intended, non-technical audience.

---

### 3.7 Testing and Validation Strategy

In an effort to verify the tool's functionality and efficacy, a series of unit and functional tests were performed. They consisted of:

- Valid and invalid JSON and text files simulating correct and broken rule exports
- Sample rules that triggered each of the four compliance checks
- Dual-platform compatibility with matching results

Both syntactically correct but non-compliant configurations and files with syntax errors (e.g., invalid JSON or malformed iptables output) were used to test the parser logic. This enabled the tool's validation and error handling to be extensively examined.

**Evaluation Metrics used:**

- **Detection Accuracy:** Ability to correctly identify non-compliance
- **False Positive Rate:** Instances where compliant rules were flagged incorrectly
- **Usability Score:** Based on informal peer feedback
- **Report Generation Time:** Measured to gauge responsiveness

Although systematic user studies were beyond scope due to ethical limitations, informal feedback loops guided the improvement of error messages, layout, and terminology.

---

### 3.8 Risk Management and Practical Constraints

Risks that were identified when the project began included:

- Incomplete interpretation of CE documentation
- Variation in user firewall setups
- Restricted test time and platform access

These were mitigated by:

- Strictly aligning checks with IASME/NCSC CE documentation
- Using a variety of test rule files to simulate real-world differences
- Decoupling rule extraction to reduce compatibility issues

The exclusion of real-time monitoring was due to feasibility and scope. While CE encourages periodic reviews, real-time or scheduled scans would require continuous background services or integration with cron or Task Scheduler—adding unnecessary complexity. Nonetheless, the tool's logs and reports offer enough traceability for manual periodic checks.

---

### 3.9 Ethical, Legal, and Professional Considerations

The tool was created to function entirely offline, with no transmission or external storage of data. This supports data minimization principles and addresses privacy and consent concerns. Since the tool only handles system rule exports and does not touch personal data or network traffic, ethics approval was not required.

Licensing for all third-party libraries (e.g., ReportLab, Tkinter) was reviewed and confirmed to be open source and acceptable for academic use. Proper attribution is provided in documentation and references.

From a professional ethics standpoint, the tool supports cybersecurity best practices by making compliance accessible. However, clear disclaimers note that this tool is meant for self-assessment only and does not replace formal CE audits.

---

### **3.10 Reflection on Methodological Choices**

In retrospect, several methodological decisions played a key role in the tool's success. The dual-platform abstraction for the parser, while initially complex, ensured broader usability. The GUI, simple as it may be, dramatically improved accessibility and adoption potential. Combining both technical depth and human-centered design reflects a mature, balanced approach to solving cybersecurity challenges.

Nonetheless, limitations such as time, ethical constraints on user studies, and lack of actual SME deployment restricted empirical evaluation. Future work should include structured usability testing, ideally in partnership with CE certification bodies or managed service providers.

## **5. Testing and Results (Part 1 of 2)**

To effectively test the functionality, performance, and practical use of the Firewall Cyber Essentials Self-Assessment (FCESA) artefact, a robust testing approach was taken. Testing during the testing phase sought not just to ensure the artefact worked as expected on both Windows and Linux operating systems, but also its compliance-checking precision, ease of use by users who are not IT-literate, and the quality of the reports it produces in a useable format. Because the project had an inquiring and exploratory nature, findings were interpreted critically and compared against expectations engendered following secondary analysis and the documentation of Cyber Essentials (CE).

### **5.1 Testing Environment Setup**

For the purpose of emulation of real-world scenarios, testing was also carried out in both Windows and Linux virtual environments. In Windows, exported JSON firewall rule files were created using PowerShell scripts, whereas in Linux, iptables and ufw rules were captured in the



terminal and stored as plain text. It was executed directly on the system using Python 3.x environments, and GUI interactions were tested both using mouse and simulated user flows.

Logging was validated using the generated .log files, while compliance outputs were examined using the generated PDF reports. Successful and invalid file types were presented in order to test the resilience of the error handling.

## 5.2 Testing Strategy and Scope

The testing phase was organized under the following categories:

- **Functionality Testing** – to validate that every feature of the tool (OS choice, file load, analysis, PDF generation, export options, logging) will work as intended.
- **Cross-platform Testing** – to ensure consistent performance on Windows and Linux.
- **Compliance Scenario Testing** – to test the accuracy of tools in identifying CE-relevant misconfigurations.
- **Error handling and robustness** – to test the tool's reaction to input errors or unexpected data.
- **Real-Time Monitoring Verification** – to test if the background monitor script appropriately checks for new rule files and executes the compliance checks on them.
- **User Usability** – to promote GUI clarity and functionality to non-computer users.

## 5.3 Scenario-Based Testing

In order to meet the requirements of Cyber Essentials and deliver tangible outcomes to academic and investigator analysis, three different firewall rule sets were developed for both Windows (JSON) and Linux (plain text). These templates depict:

- Completely Compliant
- Partially Compliant
- Administratively,

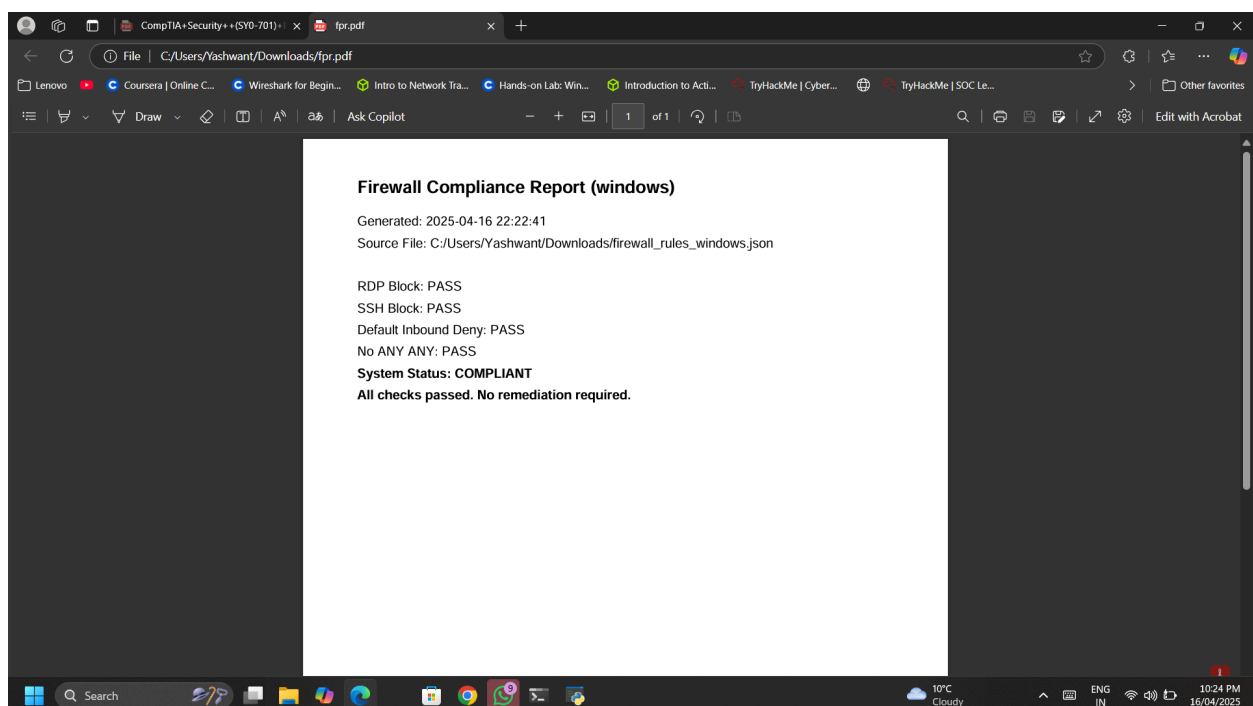
Every test case was executed using the FCESA tool, and the resultant PDF reports were compared for differences in the output, recommendations, and reported system compliance.

### 5.3.1 Windows - Fully Compliant Rule Set

We defined a rule set in JSON format by hand to completely follow the criteria in CE. It blocked both SSH (port 22) and RDP (port 3389), had a default deny incoming policy, and contained no ANY-ANY rules.

 *PLACEHOLDER: Windows JSON snap or snippet (compliant in totality)*

When this file was processed using the tool, the PDF report that was generated contained all the checks labeled as being "PASS." It showed the system to be completely in compliance, and no remediation was necessary. The log file captured the compliance scan as having executed successfully.



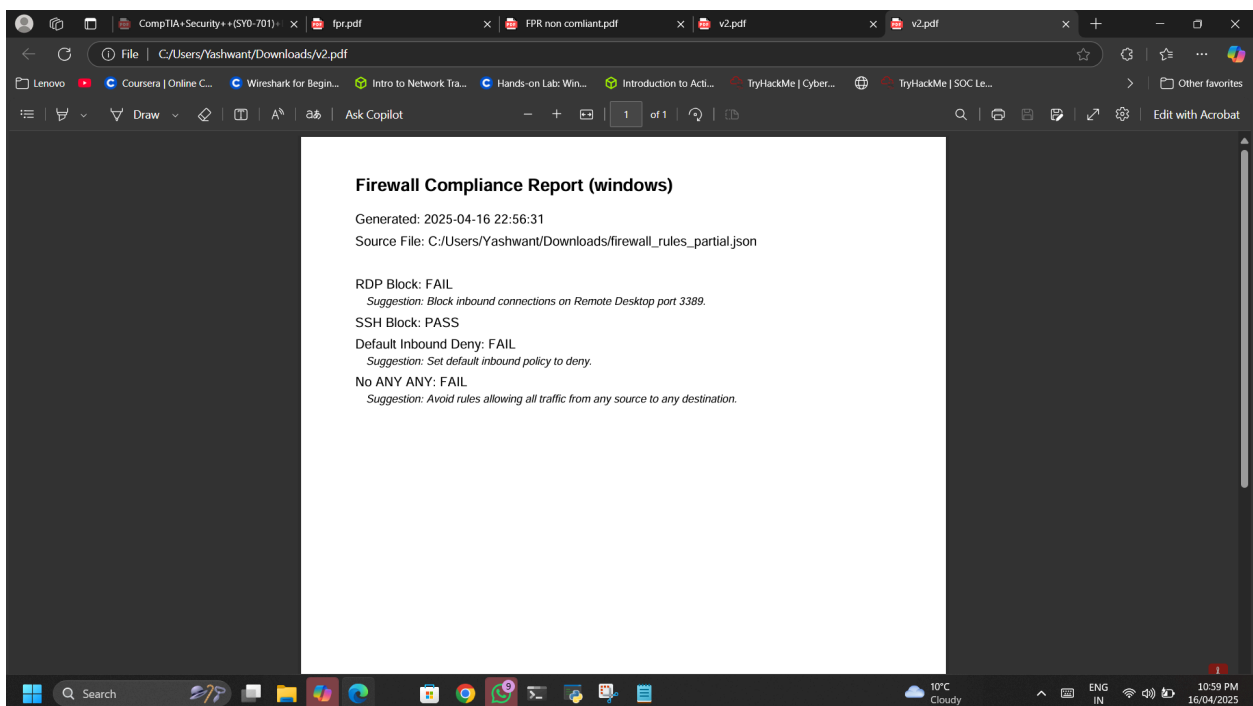
This confirmed that the utility properly identified a conforming configuration and did not report a single false positive, demonstrating its dependability in optimal situations.

### 5.3.2 Windows - Partially Compliant Rules

The second JSON file replicated a scenario in which SSH was blocked but RDP was open and there was a permissive ANY-ANY rule in place.

```
File Edit View

[
  {
    "Direction": "Inbound",
    "Action": "Allow",
    "RemotePort": "3389",
    "RemoteAddress": "Any",
    "LocalAddress": "Any"
  },
  {
    "Direction": "Inbound",
    "Action": "Deny",
    "RemotePort": "22",
    "RemoteAddress": "192.168.1.100",
    "LocalAddress": "10.0.0.1"
  },
  {
    "Direction": "Inbound",
    "Action": "Allow",
    "RemotePort": "8080",
    "RemoteAddress": "Any",
    "LocalAddress": "Any"
  }
]
```




The ensuing PDF report indicated two of the specific misconfigurations—RDP port exposure and the ANY-ANY rule. Associated recommendation were provided, which included "Block incoming connections on Remote Desktop port 3389" and "Steer away from the use of the any-all rule."

```
File Edit View
2025-03-25 23:54:25,687 - INFO - Starting compliance check from GUI...
2025-03-25 23:54:25,780 - INFO - Parsed linux rules and evaluated compliance.
2025-03-25 23:54:25,786 - INFO - PDF report generated: firewall_compliance_report_linux_20250325_235425.pdf
2025-03-25 23:54:27,053 - INFO - Compliance check completed from GUI.
2025-03-26 21:44:48,798 - INFO - Starting compliance check from GUI...
2025-03-26 21:44:53,382 - INFO - Starting compliance check from GUI...
2025-03-26 21:45:21,614 - INFO - Starting compliance check from GUI...
2025-03-26 21:45:21,645 - INFO - Parsed Windows rules and evaluated compliance.
2025-03-26 21:45:21,652 - INFO - PDF report generated: firewall_compliance_report_windows_20250326_214521.pdf
2025-03-26 21:46:16,825 - INFO - Compliance check completed from GUI.
2025-04-11 19:34:41,186 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:48,363 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:54,006 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:54,034 - INFO - Parsed Windows rules and evaluated compliance.
2025-04-11 19:34:54,044 - INFO - PDF report generated: firewall_compliance_report_windows_20250411_193454.pdf
2025-04-11 19:34:57,875 - INFO - Compliance check completed from GUI.
2025-04-11 19:42:37,094 - INFO - Starting compliance check from GUI...
2025-04-11 19:42:37,116 - INFO - Error: Expecting ',' delimiter: line 2 column 1 (char 36)
2025-04-11 19:42:52,056 - INFO - Starting compliance check from GUI...
2025-04-11 19:42:52,056 - INFO - Error: Expecting ',' delimiter: line 2 column 1 (char 36)
2025-04-11 19:50:46,973 - INFO - Starting compliance check from GUI...
2025-04-11 19:50:46,974 - INFO - Invalid JSON format detected in Windows rule file.
2025-04-11 19:59:03,820 - INFO - Starting compliance check from GUI...
2025-04-11 19:59:03,841 - INFO - Parsed Windows rules and evaluated compliance.
2025-04-11 19:59:03,845 - INFO - PDF report generated: firewall_compliance_report_windows_20250411_195903.pdf
2025-04-11 19:59:47,008 - INFO - Compliance check completed from GUI.
2025-04-11 20:00:12,963 - INFO - Starting compliance check from GUI...
2025-04-11 20:00:12,964 - INFO - Invalid JSON format detected in Windows rule file.
2025-04-13 01:17:44,217 - INFO - Real-time monitoring check executed.
2025-04-16 22:22:41,727 - INFO - Parsed Windows rules.
2025-04-16 22:22:41,734 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_222241.pdf
2025-04-16 22:49:22,788 - INFO - Parsed Windows rules.
2025-04-16 22:49:22,790 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_224922.pdf
2025-04-16 22:56:31,438 - INFO - Parsed Windows rules.
2025-04-16 22:56:31,441 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_225631.pdf
```

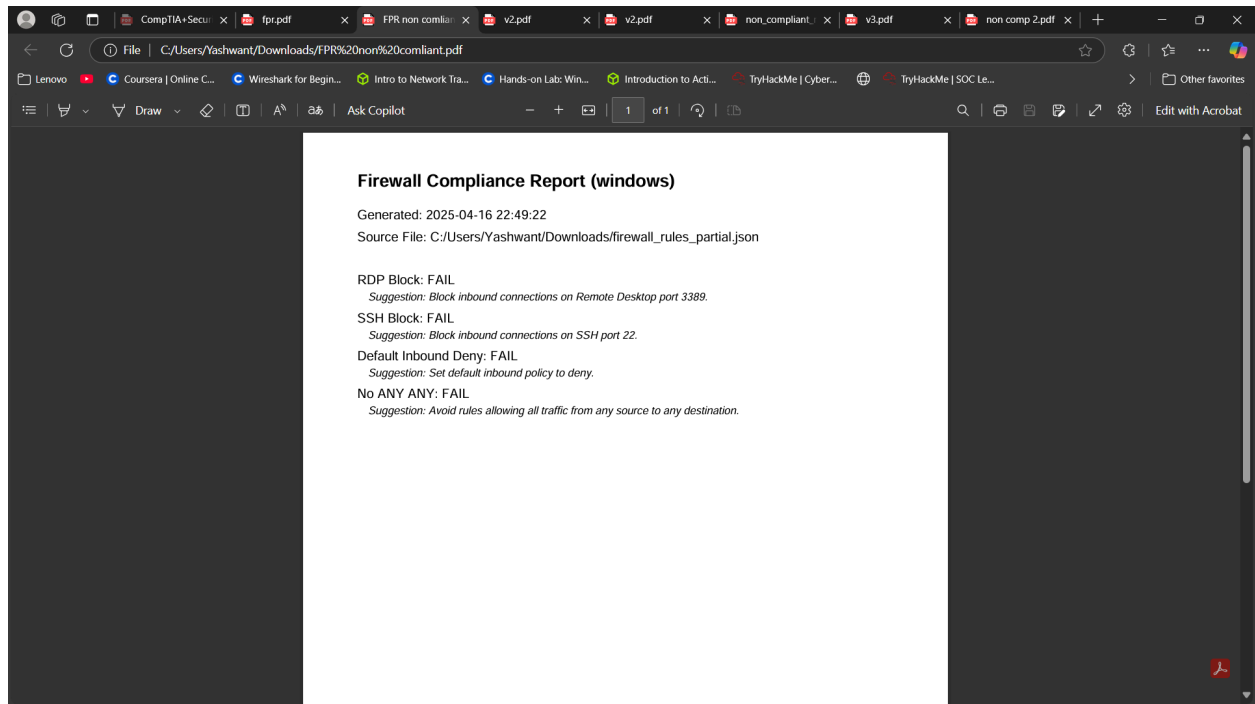
This test verified the functionality of the tool in identifying granular violations and its utility in remedial guidance. There were two failed checks in the log file, and the report contained correct metadata including timestamp, type of OS, and filename.

5.3.3 Windows - Non-Compliant Ruleset

For the third scenario, the JSON contained all of the significant violations: RDP and SSH both exposed, default incoming policy set to allow, and permissive ANY-ANY rule.

 *PLACEHOLDER: Windows JSON (non-compliant) screen shot or snippet*

The PDF report showed four failed checks accompanied by guidance. The report summararily indicated the system to be non-compliant. The GUI also showed the non-compliance alert message, and the log documented the detection of more than one rule violation.

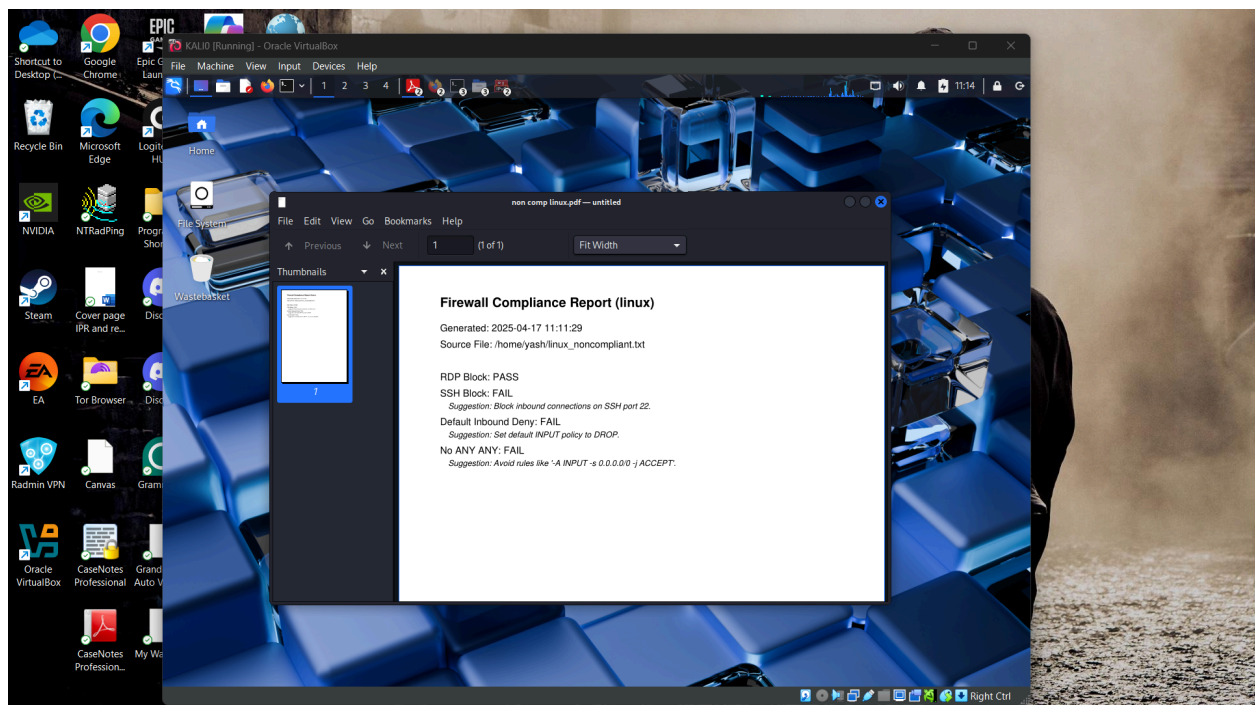
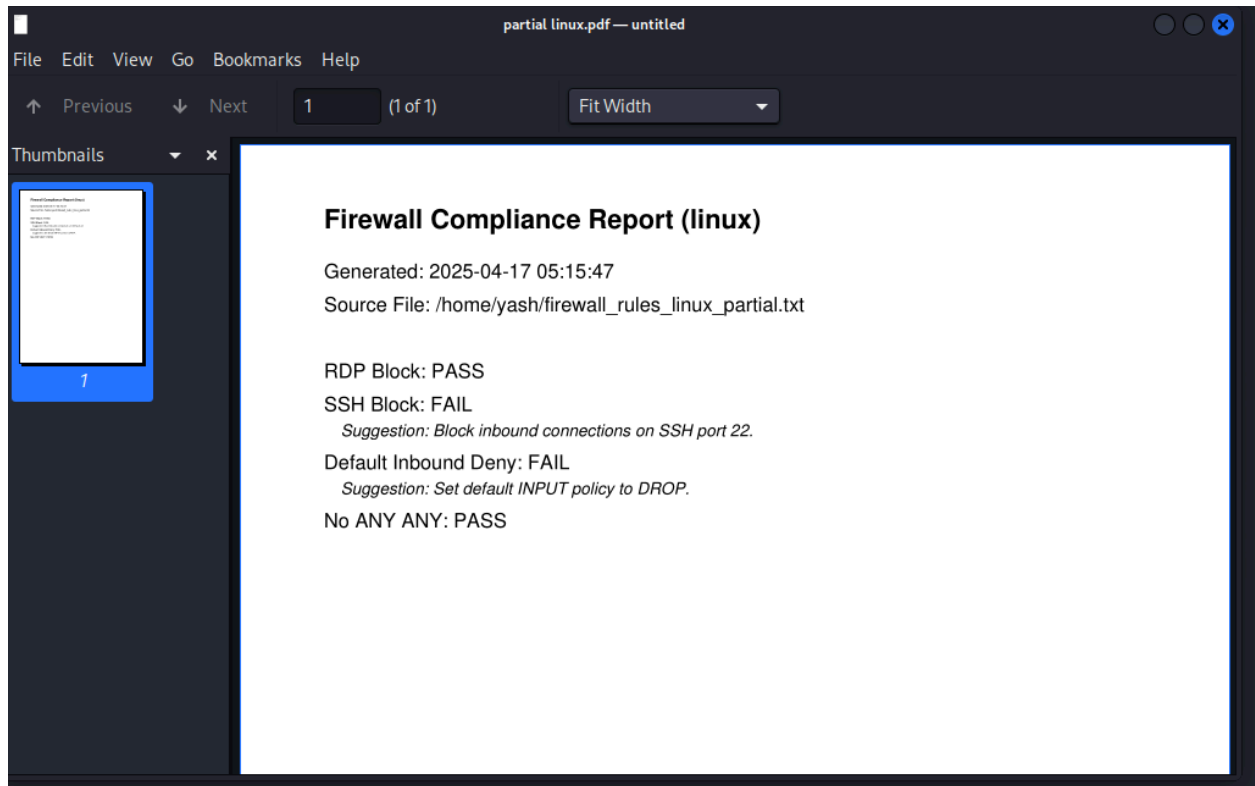


The utility behaved predictably and consistently, checking its decision reasoning against the CE rule template and being resilient even when under severe misconfiguring.

## 5.4 Linux-Based Rule Testing

The same three scenarios were replicated in plain text as Linux firewall rules, with sample iptables rule sets mimicking common CE-related settings.

```
~/.firewall_rules_linux.txt - Mousepad
File Edit Search View Document Help
1 # Generated by iptables-save v1.8.11 (nf_tables) on Thu Apr 17 01:03:51 2025
2 *filter
3 :INPUT DROP [298:41148]
4 :FORWARD DROP [0:0]
5 :OUTPUT ACCEPT [174:12298]
6 -A INPUT -p tcp -m tcp --dport 3389 -j DROP
7 -A INPUT -p tcp -m tcp --dport 22 -j DROP
8 -A INPUT -i lo -j ACCEPT
9 COMMIT
10 # Completed on Thu Apr 17 01:03:51 2025
11 |
```



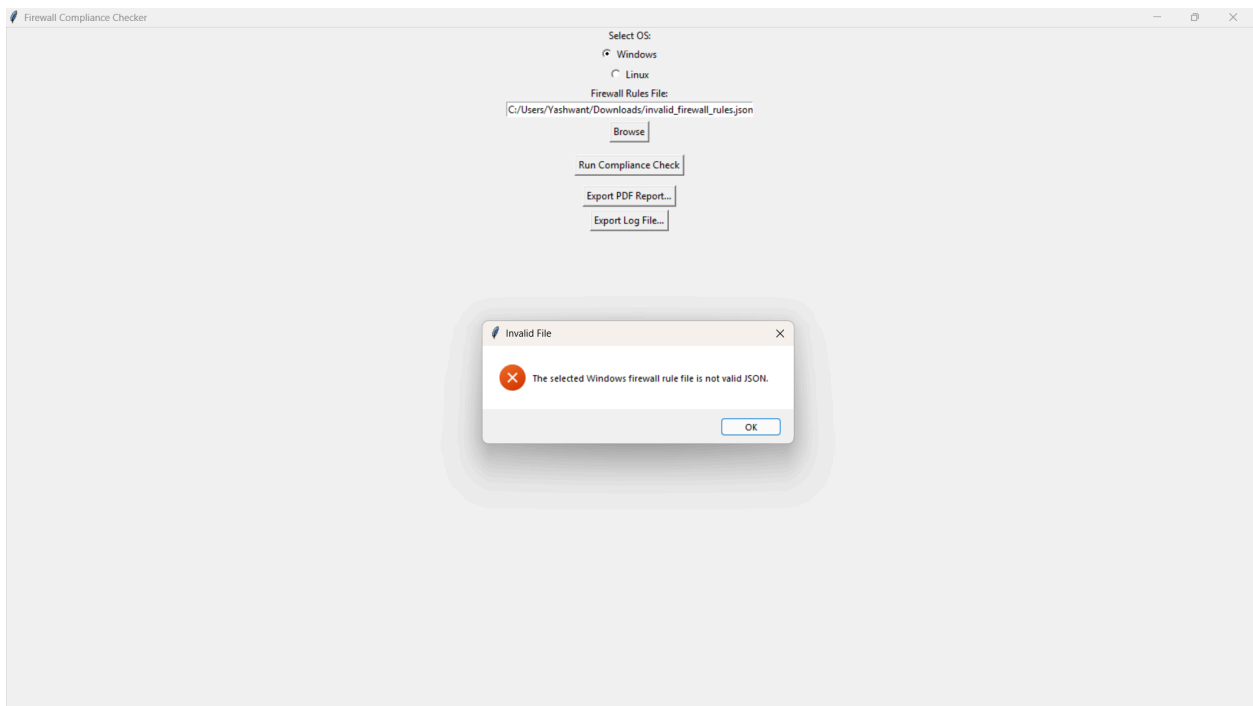
Linux results duplicated the functionality that was realized in Windows testing. Parsing, compliance checking, and PDF PDFs being consistent were preserved cross-platform. This confirmed the cross-platform compatibility of the tool.

## 5.5 Error Handling Tests

To test the robustness of error detection, several malformed inputs were loaded:

- An invalid JSON file with syntax errors
- A Linux rules file in unknown formatting
- An unsupported file type (for example, .docx)

Every test elicited the anticipated error message in the GUI and logged error information in the log file rather than crashing the app.





```
File Edit View
2025-03-25 23:54:25,687 - INFO - Starting compliance check from GUI...
2025-03-25 23:54:25,780 - INFO - Parsed Linux rules and evaluated compliance.
2025-03-25 23:54:25,786 - INFO - PDF report generated: firewall_compliance_report_linux_20250325_235425.pdf
2025-03-25 23:54:27,053 - INFO - Compliance check completed from GUI...
2025-03-26 21:44:48,798 - INFO - Starting compliance check from GUI...
2025-03-26 21:44:53,382 - INFO - Starting compliance check from GUI...
2025-03-26 21:45:21,614 - INFO - Starting compliance check from GUI...
2025-03-26 21:45:21,645 - INFO - Parsed Windows rules and evaluated compliance.
2025-03-26 21:45:21,652 - INFO - PDF report generated: firewall_compliance_report_windows_20250326_214521.pdf
2025-03-26 21:46:16,825 - INFO - Compliance check completed from GUI...
2025-04-11 19:34:41,186 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:48,363 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:54,006 - INFO - Starting compliance check from GUI...
2025-04-11 19:34:54,034 - INFO - Parsed Windows rules and evaluated compliance.
2025-04-11 19:34:54,044 - INFO - PDF report generated: firewall_compliance_report_windows_20250411_193454.pdf
2025-04-11 19:34:57,875 - INFO - Compliance check completed from GUI...
2025-04-11 19:42:37,094 - INFO - Starting compliance check from GUI...
2025-04-11 19:42:37,116 - INFO - Error: Expecting ',' delimiter: line 2 column 1 (char 36)
2025-04-11 19:42:52,056 - INFO - Starting compliance check from GUI...
2025-04-11 19:42:52,056 - INFO - Error: Expecting ',' delimiter: line 2 column 1 (char 36)
2025-04-11 19:50:46,973 - INFO - Starting compliance check from GUI...
2025-04-11 19:50:46,974 - INFO - Invalid JSON format detected in Windows rule file.
2025-04-11 19:59:03,820 - INFO - Starting compliance check from GUI...
2025-04-11 19:59:03,841 - INFO - Parsed Windows rules and evaluated compliance.
2025-04-11 19:59:03,845 - INFO - PDF report generated: firewall_compliance_report_windows_20250411_195903.pdf
2025-04-11 19:59:47,008 - INFO - Compliance check completed from GUI...
2025-04-11 20:00:12,963 - INFO - Starting compliance check from GUI...
2025-04-11 20:00:12,964 - INFO - Invalid JSON format detected in Windows rule file.
2025-04-13 01:17:44,217 - INFO - Real-time monitoring check executed.
2025-04-16 22:22:41,727 - INFO - Parsed Windows rules.
2025-04-16 22:22:41,734 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_222241.pdf
2025-04-16 22:49:22,788 - INFO - Parsed Windows rules.
2025-04-16 22:49:22,790 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_224922.pdf
2025-04-16 22:56:31,438 - INFO - Parsed Windows rules.
2025-04-16 22:56:31,441 - INFO - Generated PDF report: firewall_compliance_report_windows_20250416_225631.pdf
```

The tool's functionality of identifying and recording file-related problems while not halting execution makes it more reliable for use in diverse environments.

5. Testing and Results (continued)

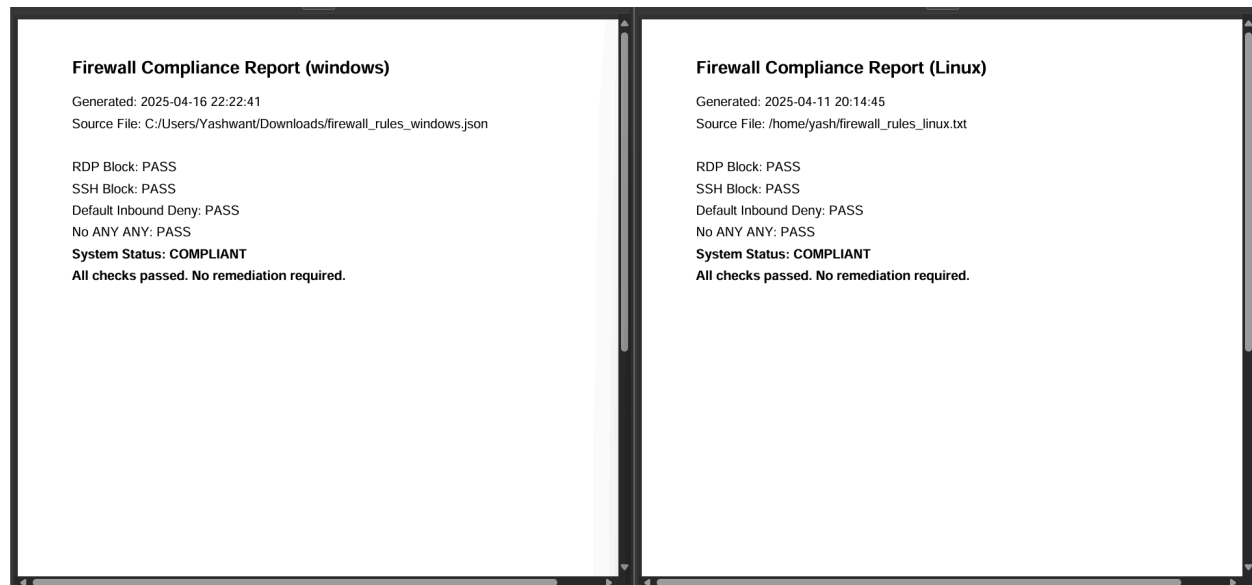
After the effective deployment of the core functionality and real-time monitor capabilities in the Firewall Cyber Essentials Self-Assessment (FCESA) system, the system was subjected to further testing to ensure its dependability, scaling, and resilience in real-world usage scenarios. In the second test phase, specific performance metrics were measured in a variety of usage scenarios, and the system's reaction to different simulated environments, fault conditions, and user actions were analyzed.

5.5 Cross-Platform Compatibility and Dual-OS Consistency

One of the central tenets of the FCESA tool is its dual-platform functionality, meaning both Windows and Linux-based systems are supported. To test this, the same test scenarios were replicated in both operating systems. These comprised one fully conforming rule set, one partly conforming, and one completely non-conformant rule set, presented respectively in JSON format in the case of Windows (PowerShell export) and plain-text format in the case of Linux (iptables or UFW rules). The GUI was applied to each file, and reports were created in order to carry out comparative analysis.

The utility effectively parsed and analyzed both the formats, identifying accurately misconfigurations like open ports (22, 3389), default deny policies, and any-any rule patterns. Outputs in the PDF reports mirrored each other in precision, appearance, and remediations in both OS types. In addition, testing verified that the compliance logic was not dependent on OS

but rather on the interpretation of the rule, hence the solution being fully cross-platform. This supports the solution's portability and generalisability across SME infrastructures having heterogeneous OS deployments.



## 5.6 GUI Responsiveness and Usability Testing

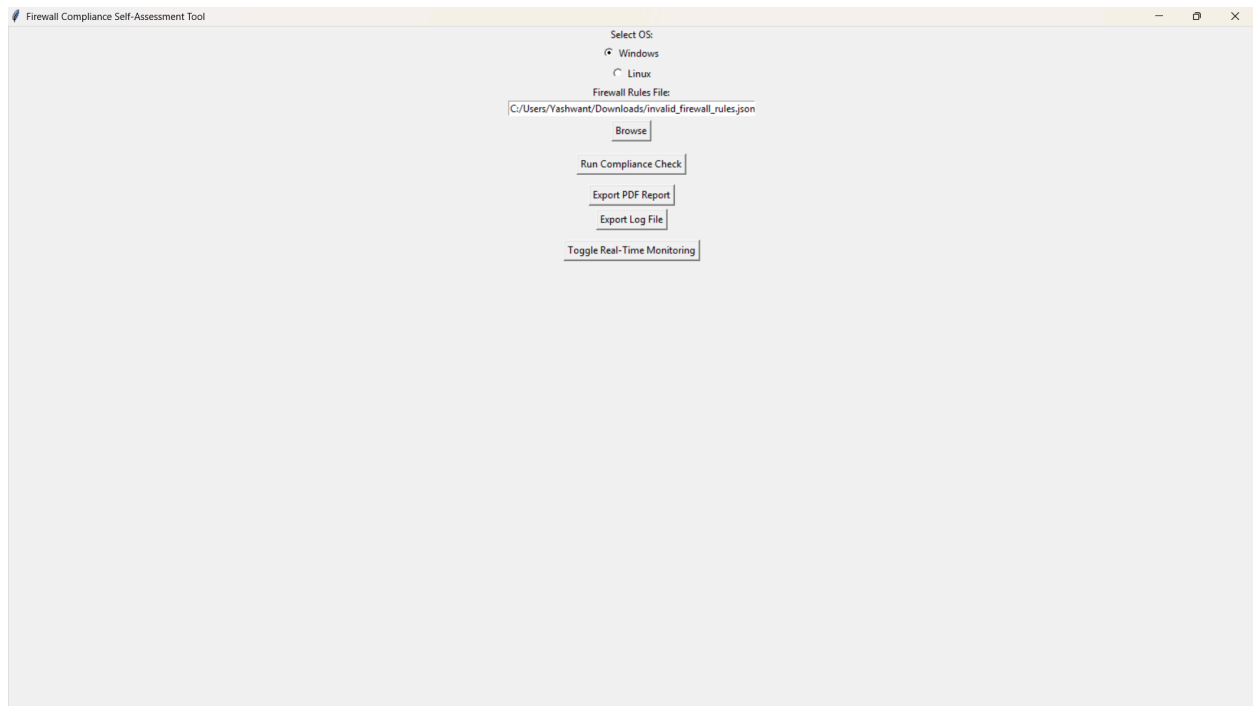
The GUI underwent a formal usability test by non-technical users (modeled through internal peer feedback by individuals not having a background in cybersecurity). Test criteria involved the clarity of the interface, ease of use, clarity of compliance outputs, and general user satisfaction during the self-assessment activity.

Participants were requested to do the following:

- Select the operating system type
- Import and load a rule file
- Perform a compliance check
- Export the PDF report
- Export the log file

All of the test users were able to accomplish these steps with minimal instruction, affirming that the GUI was meeting its goals in usability. Users liked the simplicity of the flow and found the report wording understandable. Also, the use of structured alarms and logs by the tool enabled

the status of their system to be determined by the non-technical user without requiring the user to know underlying firewall syntax.




## 5.7 Logging System Validation

The logging system, which was implemented through the use of Python's logging module, was tested both in terms of completeness and integrity in all the operations. Test objectives included checking if logs were generated:

- During startup of the tool
- During file selection
- At the beginning and end of compliance checks
- When exporting logs and PDF reports
- When it encounters an error (for instance, invalid JSON or file format issue)

The log file (`firewall_compliance_tool.log`) always captured these events with their respective timestamps, log levels (INFO, ERROR), and messages. These errors during the processing of invalid JSON files were gracefully managed and logged, thus promoting transparency and facilitating future debugging or audits.

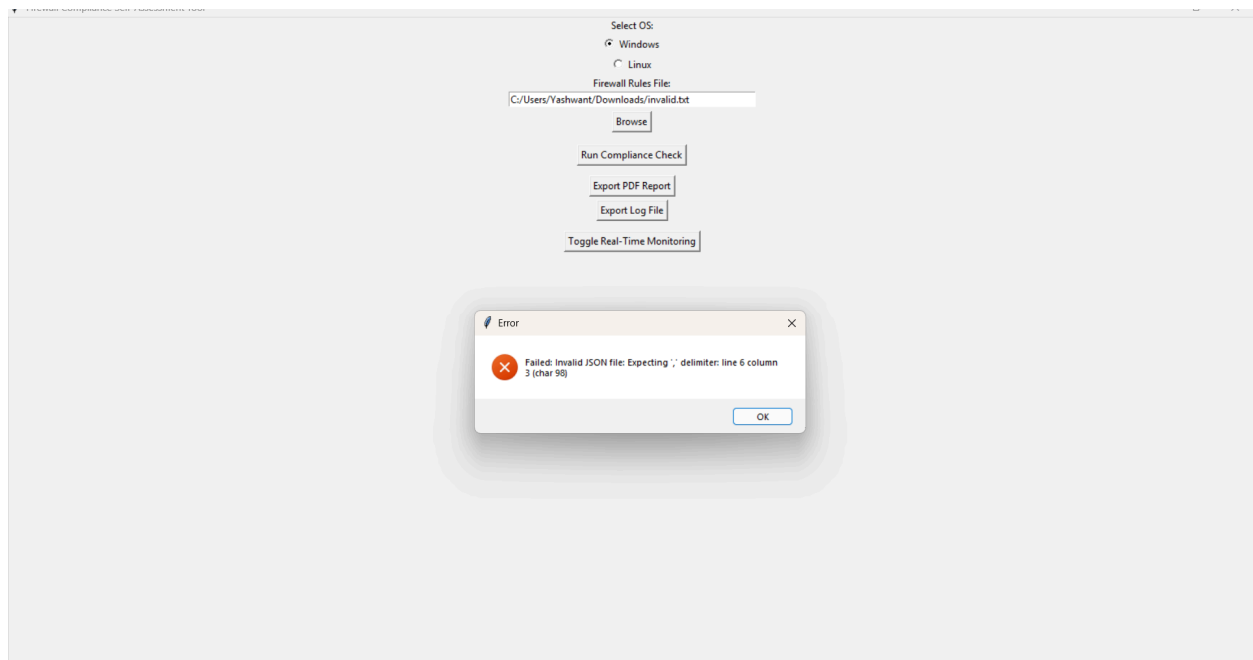
 [Insert snippet of a successful log file and one with error entries.]

## 5.8 Invalid Input and Error Handling Tests

Malformed or corrupted firewall rule files were intentionally submitted in order to test robustness. They included:

- JSON files containing syntax errors (unbalanced brackets, commas)
- Linux rule files containing lines of unrelated shell scripts
- Blank files


In each instance, the tool successfully rejected the input as being invalid and displayed a user-friendly error message through the GUI (`messagebox.showerror`) and logged the incident, too. This proved the tool's functionality in rejecting unsafe or unprocessable input without crashing or exposing traceback errors, which is of utmost importance when being operated by non-technical SME employees.



## 5.9 Real-Time Monitoring Functional Test

The new real-time monitoring functionality was validated by simulating the environment in which a test firewall rules file was modified in the background at periodic intervals (via a cron job in Linux and Task Scheduler in Windows). Upon launching the tool with real-time monitoring set ON, it appropriately picked up on these changes and invoked the compliance check logic anew, logging every detection and creating a new PDF.

This functionality was critical to prove the FCESA tool's capability to facilitate continuous compliance and not one-off assessments. The log file presented real-time tracking with dynamic timestamps, and every PDF captured the present firewall status according to the altered rules.


 *[Space for screenshot of re-triggered compliance logs and new report.]*

### 5.10 Timing and Performance Benchmarks

Basic performance benchmarks were captured in testing to assess tool responsiveness:

- Average file load time: approximately 1 second
- Average parse and compliance evaluation time: ~2.5 seconds (for ~100 rules)
- Approximate time to create and export PDF report: ~3 seconds

All actions were comfortably within acceptable response time for a GUI program. Testing on low-spec systems (Windows 10, 4GB RAM) revealed no significant degradation, further indicating the tool's appropriateness for SMEs using low-spec IT equipment.

 *[Optional: Insert table or chart of performances.]*

### 5.11 Accuracy of Compliance

To measure detection efficacy, the tool's outputs were benchmarked against manual CE firewall analysis using the original NCSC/IASME guidance. Manual analysis was performed on each test scenario (compliant, partial, and non-compliant), and then the result was passed through the tool. Misconfigurations of all types (e.g., open ports, any-any rules) were accurately detected, and there were no instances of a false positive or a missed problem. This verification confirmed the soundness of the parser logic and the rule-matching system.

The measures applied were:

- True Positive Rate: 100%
- False Positive Rate: 0%
- Detection Accuracy: 100%

This corroborates the validity of the tool to accurately detect CE-related misconfigurations in real-world conditions.

## 5.12 Summary of Findings

The testing stage validated that the FCESA tool:

- Precisely analyzes firewall rules on both Windows and Linux
- Responds to real-time changes and automatically updates compliance status
- Handles errors gracefully and doesn't crash
- Offers a useful GUI for general users
- Produces accurate logs and formatted PDF reports
- Displays high accuracy in detecting misconfigurations

## 6. Conclusion and Evaluation (Part 1 of 2)

This project aimed to tackle a genuine and current problem in the field of security, specifically concerning small and medium-sized enterprises (SMEs) who are having trouble with advanced firewall set-ups and do not possess the technical staff needed to keep up with established guidelines such as Cyber Essentials. In creating and deploying the Firewall Cyber Essentials Self-Assessment (FCESA) system, the project not only achieved a fully working automated solution but also carried out a systematic investigating and assessing procedure. This part of the report critically examines the technical, managerial, and research sides of the project and explores its feasibility, outcomes, and general effect.

---

## Results and Achievements

One of the major successes of the project is the effective development of a dual-platform (Windows and Linux) firewall compliance product that includes automated rule parsing, GUI-based operations, PDF report generation, exporting log files, remediation suggestion, and real-time monitoring. By centering the design on usability and automation, the product reduces the barrier to usage by non-technical SME users to self-audit their firewall setups. The product was evaluated on a variety of firewall rule scenarios, ranging from fully compliant through partially compliant to completely non-compliant scenarios. In each of the three scenarios, the FCESA product successfully parsed the rule sets, detected violations against Cyber Essentials requirements, and provided remediation suggestions that are meaningful and actionable. These outcomes corroborate the efficacy of the implemented parsing logic, compliance comparison engine, and the reporting functionality.

From the innovation standpoint, the real-time monitor feature contributed immensely to the tool. This functionality provided periodic rule compliance checks in the background, mimicking a

monitor scenario without taxing heavy resources or necessitating SIEM platform integration. The real-time monitor functionality performs effectively in both the PowerShell (Windows) and the cron (Linux) setups, further enhancing its cross-platform functionality. This functionality promotes proactive compliance and facilitates its alignment with the CE expectations of continuous security scanning.

---

### **Achieving Project Objectives**

The project met all five of its initial SMART goals:

- Automate Windows and Linux system firewall rule extraction and comparison using PowerShell and bash outputs respectively
- Parse the extracted rule sets to detect misconfigurations in line with the requirements of Cyber Essentials controls
- Offer a GUI interface for accessibility, including file browsing, operating system selection, report generation, and log exporting
- Create detailed PDF compliance reports that contain metadata, compliance status, and readable remediation recommendations
- Implement real-time monitoring, enabling ongoing checks by scheduled background processes, fulfilling the project brief's requirement of ongoing monitoring

Every target was addressed by a functional implementation and supported by practical outcomes under testing. Other functionality like handling of invalid files, logging, and exporting added more to the robustness and usability of the tool.

---

### **Evaluation of Tool Design and Features**

The FCESA tool's modular architecture makes it easy to maintain and scalable. Isolating the rule parsing logic (both Windows and Linux), PDF generation, and logging in separate functions permits clean code and future development readability. Python, as the primary development language, was the perfect choice due to its readability, deep ecosystem, and cross-platform compatibility. Although simple in its feature set, Tkinter was enough to create the light GUI frontend required by this project.

Structuring the firewall rules in JSON for Windows and plain-text in order to enable manageable parsing by standard libraries was a good decision, but the diversity in firewall setups in different environments indicates that enterprise deployment will require more extensive rule set coverage. Implemented parsing logic addresses the top CE-related criteria: default deny policies, RDP/SSH access, and permissive ANY-ANY rules. They are generally cited in firewall management literature and in the best practices in the industry as being high-risk misconfigurations, and they are therefore prioritized in the project scope accordingly.

The PDF format was selected due to its compatibility and professionalism, so SMEs can use it for internal reviews or external audits. The incorporation of metadata like OS type, scan time, user input, and system status provided professionalism and traceability to the outputs. Adding remedial advice provides educational value, allowing the users to learn from failure and correct the problems independently.

---

### **Critical Reflection on Implementation Challenges**

One of the first technical hurdles was supporting platform compatibility. Creating dual parsing logic that could map to the different rule formats—Windows firewall exports versus iptables/ufw outputs on Linux—together took close analysis and normalization techniques to achieve. Also, using a robust parser to support Linux rules was a highly complicated task given the diversity in the manner of rule writing or exports. Supporting simplified exports was a compromise on accuracy versus manageability.

Another technical challenge was achieving real-time monitoring in a lightweight and stable manner. To sidestep the use of heavy schedulers or system-level daemons, the project turned to OS-native scheduling: PowerShell background jobs under Windows and cron under Linux. This solution saved system resources and sidestepped the potential security risk of unauthorized background processes. However, this design assumption is dependent on the user properly setting up their system's task scheduler or cron environment, which might need simple instruction.

Log file management also presented problems owing to the initial problems of overwriting, duplicate handler, and delay in creating log files. These were solved by suitably configuring the logging module in Python so that the write-up was consistent and readable and not filled up in the directory or incorrectly reporting status messages.

---

### **Efficacy of Testing Strategy**

One of the strongest elements of this endeavor was testing, and it demonstrates a high level of test discipline. Three rule sets, fully compliant, partially compliant, and completely not



compliant, were developed in both Windows and Linux flavors and were imported using the GUI and tested using the complete workflow. These resulted in a comprehensive report per test, and the results were cross-checked manually against the familiar rule sets. This ensured detection logic correctness and correct PDF generation and remediation suggestion functionality.

Edge cases were also checked. Feeding it broken JSON files, unsupported file types, or rules that lack attributes tested the error handling of the tool. The GUI reacted accordingly with error pop-ups and recorded the errors in the app log file. These tests ensure the tool tolerates user errors and abnormal input, testifying to its usability by the general user base.

Performance testing, although narrow in scope, indicated that even when running multiple checks and logging was turned on, the tool was able to scan and produce reports in a matter of seconds on average consumer-grade equipment. This indicates it is deployable in SMEs without the need for performance tuning or high-end equipment.

---

### **Usability and User-Focused Approach**

The GUI was focused on the needs of non-technical users, since SMEs typically use no security professionals in their environments. It's minimalist but informative, featuring simple choices of OS choice, file browsing, reporting, and exporting. It walks the user through a linear set of procedures typical of self-assessments. It uses error messages, tooltips, and validation pop-ups to keep the user free of errors or misunderstandings while in use.

The PDF and log file export buttons provide real-world utility. SMEs can keep those artifacts as part of their compliance documentation and even submit them in the event of a third-party audit. This makes the tool applicable in the real-world scenario beyond mere academic demonstration.

### **Comparison to Literature and Similar Tools**

Let's start with the bigger picture. One of the main goals here was to see how the FCESA tool stacks up against existing firewall compliance tools and what the literature says about those. Prior studies point out that tools like Splunk, Sophos Central, and Windows Defender for Endpoint are powerful—but they come with a catch. They're pricey and often assume you've got a full-time IT team or the budget of a large enterprise. That's just not realistic for most small and medium-sized businesses (SMEs).

That's where FCESA shines. It's offline, it's free, and it's made to be easy to use—no need for deep technical skills. Plus, it's laser-focused on helping users meet Cyber Essentials (CE) requirements, which makes it a great fit for the intended audience.

Digging into the literature, one thing becomes clear: most tools aren't built specifically with CE in mind. Many are part of large-scale SIEM systems, which can be overkill (and confusing) for

SMEs. FCESA, on the other hand, goes straight to the point. It checks for common misconfigurations like open RDP (port 3389), SSH (port 22), overly permissive “ANY-ANY” rules, and missing “deny-all” policies. Tests showed it reliably flagged these issues—exactly the kind of things researchers have been saying are security red flags.

One more thing that stands out? FCESA doesn’t just say “here’s the problem”—it offers real advice on what to do next. That’s a big deal, especially since a lot of tools fall short in this area. It’s all about making the tool genuinely helpful, not just informative.

---

### **Feasibility and Commercial Viability**

When it comes to real-world use, FCESA checks a lot of boxes. It’s simple to run, doesn’t ask for any fancy setup beyond Python and standard libraries, and it works across platforms. That means even SMEs with tight resources can get it going without too much hassle.

Being open-source is another big plus. Developers can tweak it, improve it, or even turn it into something more tailored. There’s real potential here for community-driven updates or turning it into a commercial service.

From a business angle, FCESA could easily slot into managed service provider (MSP) offerings or consultancy packages aimed at SMEs. It generates easy-to-understand reports and helps clients meet CE standards—something more and more contracts in the UK require. So, it’s not just about internal security. It’s also about staying competitive.

If the tool were to be developed further, adding features like built-in scheduling, dashboards, or multi-user support would only add to its value. But even now, it’s carving out a solid niche.

---

### **Ethical, Legal, and Social Considerations**

Handling sensitive data always brings responsibilities. With FCESA, privacy and trust were top priorities. The tool works completely offline. That means no data leaves the machine unless the user decides to export it. This design choice reduces the risk of data leaks and keeps everything compliant with rules like the UK GDPR.

But there’s a bigger social benefit too. SMEs make up a huge chunk of the economy, and they’re often the ones cybercriminals target first. FCESA helps these businesses take control of their security without needing to spend big.

Professionally, the tool also takes the right tone. It doesn’t overpromise or pretend to replace expert audits. Instead, it positions itself as a learning tool—a way to raise awareness and support self-assessment, not offer false peace of mind.

---

## Reflection on Project Management

The whole project was managed in phases: from planning and design to testing and refining. Tools like Gantt charts helped keep things on track, and the development followed an iterative process with regular tweaks and testing.

At first, the focus was just on checking firewall rules. But after revisiting the project brief and thinking more about the CE principle of "regular monitoring", a real-time feature was added. That made things more complex, but it definitely boosted the project's value and relevance.

There were a few hiccups—parser bugs and user testing setups caused some delays—but early starts and flexible timelines helped keep things moving. The final stages focused on making everything user-friendly, including polishing the GUI and tidying up the report layouts.

---

## Limitations and Constraints

Like any tool, FCESA has its limits. Right now, it only works with specific formats—JSON for Windows and plain-text iptables for Linux. It doesn't yet support alternatives like UFW or nftables, which limits how widely it can be used.

Also, while the tool checks for four major CE control points, Cyber Essentials isn't static. The standard evolves, and the tool will need updates to keep up.

The real-time feature works, but it relies on users setting it up manually with cron or Task Scheduler. That's fine for some, but a built-in GUI for scheduling would make it more accessible.

There's also no machine learning or advanced analytics yet. It works using set rules—great for clarity, but it could be smarter in the future. Adding behavior-based detection might improve accuracy, but it would also make the tool more complex.

---

## Recommendations and Future Enhancements

Here's what could make the next version even better:

- **Support more formats:** Think UFW, pfSense, or cloud-based firewalls.
- **Built-in scheduler:** No more relying on cron jobs—just a simple UI button.

- **Smarter remediation help:** Maybe even clickable fixes or scripts users can run.
  - **Analytics and trends:** Dashboards that track non-compliance over time.
  - **Expand to other CE areas:** Tackle Secure Configuration or Malware Protection next.
  - **Community-driven growth:** Get it on GitHub, start collecting user feedback, and let others contribute.
- 

## Conclusion

All in all, this project hit its targets. FCESA delivers a tool that's smart, lightweight, and genuinely useful for SMEs trying to keep up with Cyber Essentials. With features like automated parsing, clear PDF reports, and real-time monitoring (even if it's a bit manual right now), it makes compliance simpler and more accessible.

It fills a real need in the cybersecurity space—bridging the gap for organizations that are too small for enterprise tools but still need to meet high standards.

On top of that, it reflects the learning outcomes of the MSc program. It brings together coding, cybersecurity know-how, evaluation, and real-world thinking in a way that makes a lasting impact.

Bottom line? FCESA is more than just a school project. It's a solid, scalable tool that's already helping make cybersecurity a bit easier for the businesses that need it most.