© March 2026 | IJIRT | Volume 12 Issue 10 | ISSN: 2349-6002

DEFENDERPEAS - Privilege Escalation Assessment and Safeguard Tool

Mr. D Matru[1], K. Manideep[2], K. Saikrishna[3], M. Shiva[4], K. Ranadhir[5]

[1]*Assistant professor, Department of Computer Science and Engineering Cyber security Sphoorthy Engineering College.*

[2,3,4,5]*Student, Department of Computer Science and Engineering Cyber security, Sphoorthy Engineering College.*

Abstract—**This paper presents Defender PEAS, a platform that helps find and prevent attacks that try to gain access on a computer system. Defender PEAS is designed for companies and models how attacks can happen across different system boundaries. Traditional tools just list what is wrong with a system. Defender peas do more. It uses a structure with many layers to turn system data into useful security information. The framework has parts that work together. Some parts gather information about the system without changing it. Another part takes this information. Makes it structured and trustworthy. A graph engine then shows how different parts of the system are connected. Defender PEAS then uses rules and checks to find paths of attack. It looks at transitions like user to administrator, administrator to system. From a container to the host. This helps in understanding attacks. The architecture of Defender PEAS makes sure that each part has a job. One part gathers information. Another part makes sense of this information. A third part analyses connection. Tests show that Defender PEAS is better at finding and understanding attacks than tools. It provides outputs that can be used in security dashboards. This helps in assessing and fixing vulnerabilities, in company environments. Defender PEAS helps companies protect themselves proactively.**

Index Terms—**Attack path analysis, graph-based security modeling, privilege escalation detection, System security analysis, Trust relationship modeling.**

I. INTRODUCTION

Privilege escalation is a security threat in today's computing. Attackers often use system mistakes, weak permissions. Wrongly set up services to get more control. When they get privileges, they can access sensitive data, change system operations and stay in the system for a long time. So, finding and stopping these escalation paths is crucial for keeping systems secure and organizations safe. Traditional tools for checking privilege escalation usually look for known problems or misconfigurations. They make lists of potential issues but don't give much context on how to use these weaknesses to escalate privileges. Security analysts have to go through these findings to see if a reported issue can actually lead to more control. This manual process takes a lot of time can be wrong and is hard to manage in systems.

To solve these problems, we introduce Defender PEAS. It's a framework that models system boundaries to find and analyze potential privilege escalation chains. Of just listing system mistakes Defender PEAS models relationships between system parts and checks how these can be misused to cross privilege lines. This data is used to build a graph of system trust relationships. This helps find attack paths. The system then uses rules, intelligence and risk scoring to determine how feasible and severe detected privilege escalation chains are. By modeling transitions like user to administrator or service to system Defender PEAS gives an understanding of how attackers may exploit system trust. The architecture of Defender PEAS separates system data collection, analysis and scoring into parts. It also reduces alarms. The system produces structured outputs that can be used in security dashboards and automated workflows. The main goal of this research is to create a platform that detects privilege escalation paths, through structural analysis. By combining graph-based modeling with intelligence-assisted analysis Defender PEAS aims to improve security assessments. It helps organizations proactively find and mitigate privilege escalation risks.

II. RELATED WORK

Cybersecurity is a deal. People have been studying how to catch privilege escalation and make sure systems are secure. People have created a lot of tools to find weaknesses and mistakes that bad guys can use to get power on a system. These tools are pretty good at finding things like file permissions that are not set right settings, weak login systems and old software that needs to be updated. Most of these tools look for known problems. The problem with these tools is that they are not good at understanding how all these weaknesses and mistakes are connected and how bad guys can really use them to attack a system.

The tools are good, at finding problems. They do not really understand how bad guys can use these problems to get power on a system. There is a tool called Lin PEAS that is used a lot to check for privilege escalation on Linux systems. It looks at all the system settings, like file permissions, running services, environment variables and scheduled tasks. It finds problems based on known patterns and mistakes. Even though this tool is really good at gathering information it just makes a big list of problems and does not automatically figure out how they can be used together to really attack the system. There is another tool called Win PEAS that is used on Windows systems. It checks registry entries, service permissions, startup settings and scheduled tasks for weaknesses. These tools help people who test system security and do research. They need analysis to determine if a vulnerability can be used to gain higher privileges. The tools provide results that need to be examined to see if a problem can escalate privileges. Security testers and researchers rely on these tools to find security issues. They help identify weaknesses that could be used to gain control. The goal is to find vulnerabilities that can be used to escalate privileges and compromise system security. These tools are useful, for people who want to test and improve system security.

They help. Analyze potential security problems. The results need to be reviewed to determine the course of action. Security testing and research require judgment and analysis. The tools provide information but human expertise is needed to interpret the results. There is also a framework called Metasploit Framework that has tools for privilege escalation and exploiting vulnerabilities. It is really powerful. It focuses more on exploiting weaknesses than on understanding how the system really works. So, people still have to use their reasoning to understand how different problems interact with each other on the system. Some new research has been looking at using graph-based models to show how system parts are connected and how privileges work. In these models' system parts are like nodes and trust relationships and permission flows are like edges. This makes it possible for computers to automatically figure out attack paths and privilege escalation chains. These models are really good at finding attack scenarios that involve many connected problems. Even with all these advancements many tools just focus on finding problems or on exploiting them rather than providing a structured way to collect system data, model trust relationships correlate attack paths and evaluate risk. Most traditional tools do not even model the boundaries between levels of privilege like from user to administrator which is really important for understanding real-world escalation scenarios. The Defender PEAS framework is trying to fix these problems by introducing an architecture that separates system data collection, normalization, graph-based correlation, intelligence-driven evaluation eligibility analysis and risk scoring. By turning raw system information into trust data and modeling relationships, with a graph engine the framework can automatically find realistic privilege escalation chains. This makes it easier for security analysts to understand the results and reduces the amount of work they have to do to assess system security. Defender PEAS is a way to check system security. It uses a graph engine to model system relationships. It can automatically find privilege escalation chains. It makes it easier for security analysts to do their job. Another used tool for checking system security is OpenVAS.

It scans networks to find weaknesses in systems and services. OpenVAS provides reports that show known vulnerabilities and configuration issues. These reports help identify security problems. However, tools like OpenVAS usually focus on finding known security flaws. They do not analyse how multiple weaknesses can combine to gain access within a system. Nessus is another tool used by big companies to scan for security issues. It checks operating systems, applications and network services for vulnerabilities. Nessus does this by comparing system settings against a database of known vulnerabilities. These scanners are good at finding vulnerabilities. They often produce long reports with many findings that need manual review. This review helps determine the impact on system security and access controls. Vulnerability scanners like OpenVAS and Nessus are essential, for identifying security weaknesses. They help companies prioritize and fix security issues. way communication without dependence on intermediaries, such systems promote inclusivity, support equal employment opportunities, and strengthen confidence in professional interview environments.

III. SYSTEM DESIGN AND METHODOLOGY

1. Problem Definition:

Modern operating systems are really complicated. They have a lot of parts that interact with each other like users, services, files, processes and kernel components. A lot of the time security problems do not come from one flaw but from a chain of trust relationships that let an attacker move across security boundaries. Traditional tools that look for security problems usually look at each part of the system separately. Make a list of potential weaknesses.

These tools do not often look at how all the different parts of the system work together to create a path that an attacker can use. The main problem that the Defender PEAS system tries to solve is: how to understand the trust relationships in an operating system and figure out if those relationships let an attacker cross security boundary by using multiple steps. To solve this problem the system uses a structured framework that is based on trust primitives graph modelling and attack path analysis.

2. Threat Model:

The system assumes that there is an attacker who has some privileges on a host system.

The Defender PEAS system thinks about what the attacker can do such as:

- Running commands as a privileged user
- Writing to files or directories that they have access to
- Changing user-controlled configuration files
- Triggering scheduled processes or services

The attacker does not have kernel privileges at first. The attacker wants to get privileges like becoming an Administrator, SYSTEM, root, kernel or getting domain administrative privileges. The system assumes that the operating system is working normally and that the attacker is using misconfigurations or insecure trust relationships to exploit the system.

3. Core Architectural Philosophy:

The architecture of the Defender PEAS system is based on some design principles. These principles include:

Layered Security Analysis: the system breaks down security analysis into layers each with its own responsibility Structural Modelling: the system looks at the relationships between parts of the system rather than just individual configuration values

Privilege Boundary Awareness: the system understands the transitions between levels of privileges

Deterministic Analysis Pipeline: each stage of analysis changes the data in a controlled and predictable way

Separation of Concerns: the system keeps different parts of the analysis separate, like data collection, modelling, intelligence and scoring

4. System Architecture Overview

The Defender PEAS system is made up of a series of analysis stages. These stages include:

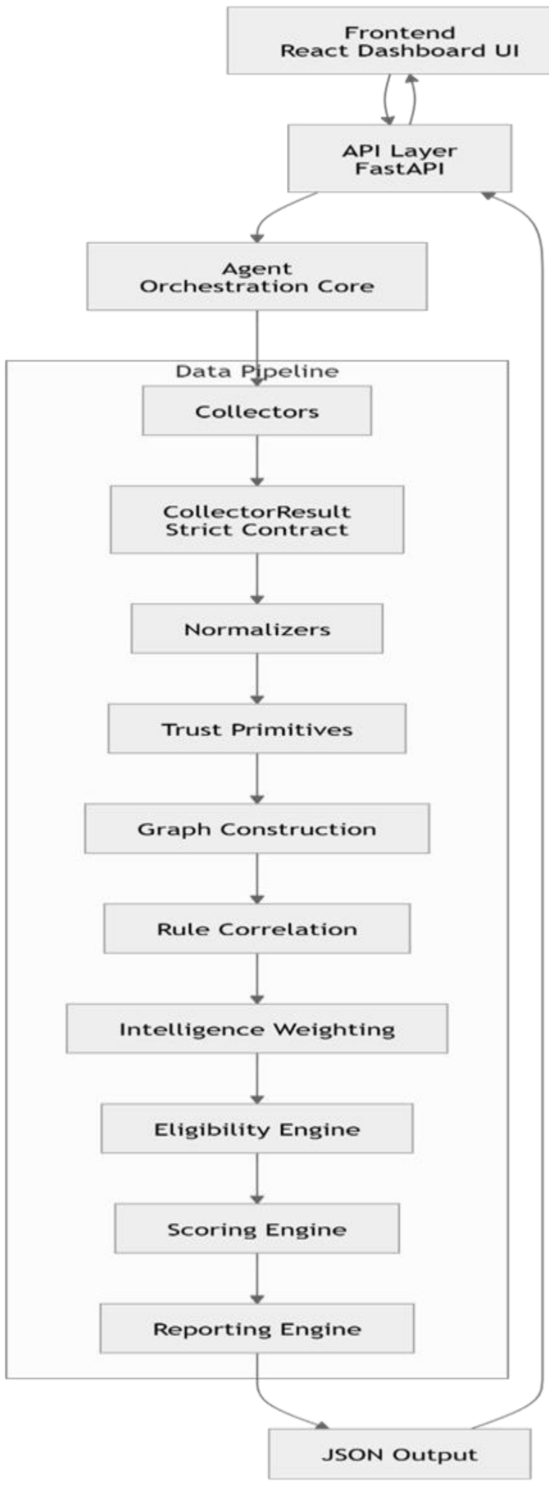


Fig1: system architecture

State Collection Layer: collects configuration and runtime information from the host system
Normalization Layer: converts data into a standardized format

Trust Primitive Modelling: creates trust primitives that represent relationships between entities

Trust Graph Construction: builds a graph of trust relationships

Correlation and Pattern Detection: looks for patterns that could be used in an attack

Intelligence Augmentation: adds knowledge to the analysis

Exploit Eligibility Analysis: determines if an attack path is realistic

Risk Scoring: assigns a risk score to each attack path
Reporting and Visualization: creates reports and visualizations of the results

5. System State Collection
The state collection stage collects information about the host system. This includes things like:

- Service configurations
- Scheduled tasks
- File permissions
- Registry keys
- Environment variables
- User privileges
- Installed drivers
- Kernel modules

The collectors are designed to observe the system without changing anything. They gather information about the system state without trying to interpret what it means for security.

6. Data Normalization
The raw data from the system is often not consistent. The normalization stage converts this data into a format that can be analysed. This stage creates objects that represent meaningful security relationships. For example, of just storing file permission metadata the system creates statements like:

- User has written access to file
- Service executes binary
- Process loads library

This makes it easier for the analysis engine to understand the security relationships in the system.

7. Trust Primitive Formalization
The core concept in the Defender PEAS system is the trust primitive. A trust primitive represents a relationship between two entities that could affect privilege flow. A trust primitive can be represented as:
Subject → Relationship → Object

For example:
User → write → file
Service → execute → binary
Process → load → library
Kernel → trust → driver

Each trust primitive describes a way that privileges could propagate.

8. Trust Graph Construction
The trust primitives are used to build a directed graph of trust relationships. In this graph nodes represent entities like users, services, processes, files, drivers or containers. Edges represent trust relationships. This graph models the system as a network of privilege propagation pathways.

For example:
User → file
Writable file → executed by service
Service → runs as SYSTEM

This sequence forms an escalation chain. The graph modelling allows the system to identify multi-step attack paths that would not be visible otherwise.

9. Attack Path Correlation
Once the trust graph is built the system looks for patterns that could be used in an attack. These patterns represent known privilege escalation strategies, like:

- Writable executable paths used by services
- User-controlled scripts executed by scheduled tasks
- Unrestricted driver loading capabilities
- Dynamic library search order manipulation

The correlation engine analyses the graph to see if these patterns exist.

10. Privilege Boundary Formalization
A key innovation of the Defender PEAS system is the modelling of privilege boundaries. A privilege boundary represents a transition between two security domains. Examples include:
User → Administrator
Administrator → SYSTEM
User → root
root → kernel
Container → host
Service account → domain administrator

The system evaluates whether detected attack paths cross one or more privilege boundaries. If a path does not cross a boundary, it may represent a misconfiguration. It does not necessarily result in privilege escalation.

11. Intelligence Augmentation
The intelligence layer adds knowledge to the analysis. This knowledge includes things like:

- privilege escalation techniques
- Contextual risk modifiers
- Configuration heuristics
- Mitigation mappings

The intelligence does not change the relationships in the trust graph. Instead, it enhances the interpretation by assigning weights or confidence values to detected conditions.

12. Exploit Eligibility Analysis
Not every potential attack path is realistic. The eligibility analysis stage evaluates whether the conditions required for exploitation are satisfied. This considers factors like:

- ➢ Required permissions
- ➢ Process execution conditions
- System configuration constraints
- Runtime dependencies

The eligibility analysis reduces the likelihood of positives by filtering out unrealistic attack paths.

13. Risk Scoring Framework
Each detected attack path is assigned a risk score. The scoring model combines two metrics:

- Severity: the impact if the attack path is successfully exploited
- Confidence: the likelihood that the attack path can be executed in the environment

These metrics are combined to produce an overall risk level classification. This scoring framework allows defenders to prioritize remediation efforts.

14. Output Generation

The final stage produces security reports. These reports include things like:

Identified abuse patterns

- Attack path descriptions
- Privilege boundaries crossed
- Risk severity levels
- Recommended mitigation actions

The output is provided in a format that can be consumed by APIs, dashboards or security management platforms.

15. Methodological Contributions

The Defender PEAS system introduces methodological improvements compared to traditional vulnerability scanners. These include:

Attack Path Modelling: security weaknesses are analysed as part of multi-step exploitation chains

Trust Relationship Analysis: system security is evaluated through interactions between entities

Privilege Boundary Awareness: escalation is defined by transitions, between security domains

Modular Architecture: the layered pipeline allows each stage to evolve independently

Reduced False Positives: eligibility analysis improves accuracy by evaluating exploit feasibility

16. Conceptual Outcome

The Defender PEAS methodology transforms system configuration data into a representation of privilege propagation pathways. By combining trust primitives graph analysis, privilege boundary modelling and exploit eligibility evaluation the system provides an understanding of how privilege escalation may occur within a system. Of just identifying insecure configurations the system reveals how those configurations could be combined into realistic attack chains. individual alphabet characters and retrieves corresponding alphabet-level ISL videos.

Matched video segments are concatenated sequentially using frame-wise merging. Frame dimensions are standardized (e.g., 640 x 480 pixels) to ensure uniform playback. The final video is encoded into a browser-compatible format for rendering in the user interface. This hierarchical mapping strategy ensures semantic preservation while gracefully handling vocabulary limitations.

IV. IMPLEMENTATION

The proposed system was built based on the design described earlier. The system is made up of independent modules each handling a specific part of the analysis process. The main goal of the system is to bring different parts into a working pipeline. This pipeline collects data from the system analyses trust relationships and provides security insights. The backend analysis engine is in charge of managing the process. It starts the data collection processes the information and analysis it to find security risks. The data collectors are modules that gather system information. They collect details about system services, tasks, file permissions and user privileges. The collected data is then organized in a format. The data is then converted into a security format that shows trust relationships between system parts. This format is used to create a trust graph that shows how users, services, files and processes interact. The system then uses graph analysis to find patterns that could be security risks. It checks these patterns to see if they could lead to security breaches. The analysis results are then turned into reports. These reports include detected security risks, severity levels and confidence indicators. The results can be accessed through an API. Viewed on a user interface.

The systems modular design allows parts to be easily changed or added without affecting the system. This makes it flexible and ready for improvements.

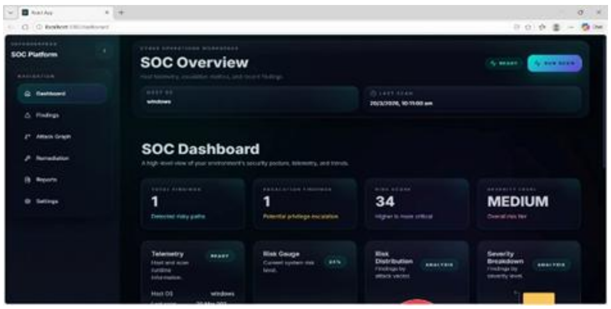


Fig 2: Dashboard

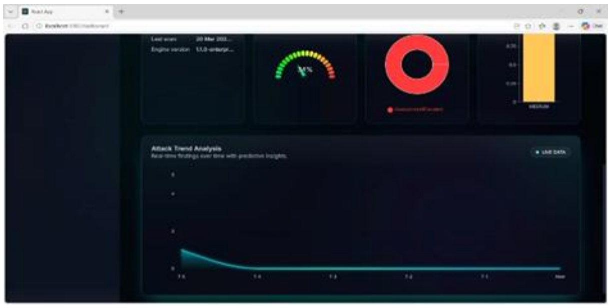
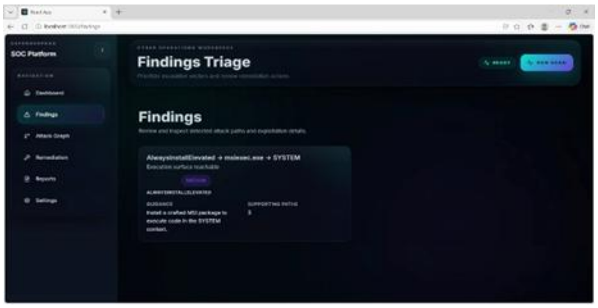


Fig 3: attack trend analysis



Install a crafted MSI package to execute code in the SYSTEM context.

Fig 4: Findings

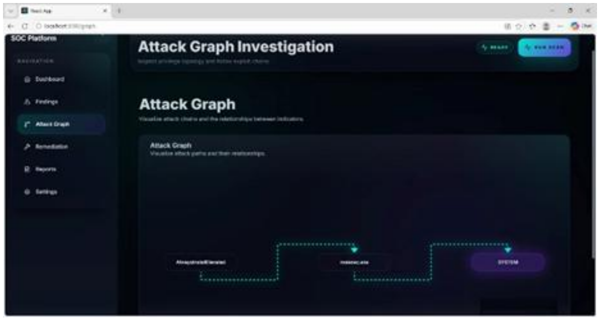


Fig 5: Attack Graph

V. FURTHER SCOPE

There are ways to improve the system. One idea is to add machine learning to help find security risks. This could help find risks that were not known before. Another idea is to make the system work with operating systems and environments. This would make it useful in situations. The systems visualization could also be improved. This could include graphs that let security analysts explore detected risks. Future work could also include real-time monitoring.

VI. CONCLUSION

The Defender PEAS system provides a way to analysed operating system security. It models trust relationships. Finds potential security risks. Unlike vulnerability scanners this system analysis interactions between system parts. It determines how multiple relationships can combine to create security risks. By representing system configurations as trust primitives and constructing a graph-based model the system detects security risks.

REFERNCES

[1] WinPEAS and LinPEAS
[2] Mitigation of Privilege Escalation Attack using
[3] Kernel Data Relocation Mechanism (KDRM)
[4] Authors: Hiroki Kuzuno, Toshihiro Yamauchi Year: 2024
[5] Detection of Hidden Privilege Escalations in Android Authors: M.A. El-Zawawi et al. Year: 2025
[6] MITRE Corporation. "Common Weakness Enumeration (CWE). https://cwe.mitre.org
[7] MITRE Corporation. "ATT&CK Framework." https://attack.mitre.org
[8] Bishop, M. Computer Security: Science. Addison-Wesley, 2018.
[9] Anderson, R. Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley, 2020.
[10] Stallings, W. Network Security Essentials: Applications and Standards. Pearson.
[11] OWASP Foundation. "OWASP Top Ten Security Risks." https://owasp.org
[12] Ammann, P., Wijesekera, D., & Kaushik, S. " Graph-Based Network Vulnerability Analysis." IEEE Symposium on Security and Privacy.
[13] Sheiner O. Et al. "Automated. Analysis of Attack Graphs." IEEE Symposium, on Security and Privacy.
[14] National Institute of Standards and Technology (NIST).
[15] Guide to Enterprise Patch Management Technologies.

ABOUT THE AUTHORS

Mr. Matru Dhanavath,
M. Tech, (Ph.D.)
Assistant professor, Department of Computer Science and Engineering- Cyber security,
Sphoorthy Engineering College.

K. Manideep, Student,
Department of Computer Science and Engineering-Cyber security, Sphoorthy Engineering College.

K. Saikrishna,
Student, Department of Computer Science and Engineering-Cyber security,
Sphoorthy Engineering College.

K. Ranadhir, Student,
Department of Computer Science and Engineering-Cyber security,
Sphoorthy Engineering College.