Scientific Research Publishing

Journal of Software Engineering and Applications, 2024, 17, 487-507
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Cybersecurity Risk Management through Behavior-Based Contextual Analysis of Online Logs

Irawati Edlabadkar, Vijay K. Madisetti

School of Cybersecurity & Privacy, Georgia Institute of Technology, Atlanta, GA, USA
Email: iedlabadkar3@gatech.edu, vkm@gatech.edu

How to cite this paper: Edlabadkar, I. and Madisetti, V.K. (2024) Cybersecurity Risk Management through Behavior-Based Contextual Analysis of Online Logs. *Journal of Software Engineering and Applications*, **17**, 487-507.
https://doi.org/10.4236/jsea.2024.176027

Received: April 29, 2024
Accepted: June 4, 2024
Published: June 7, 2024

Abstract

This paper studies cyber risk management by integrating contextual log analysis with User and Entity Behavior Analytics (UEBA). Leveraging Python scripting and PostgreSQL database management, the solution enriches log data with contextual and behavioral information from Linux system logs and semantic datasets. By incorporating Common Vulnerability Scoring System (CVSS) metrics and customized risk scoring algorithms, the system calculates Insider Threat scores to identify potential security breaches. The integration of contextual log analysis and UEBA [1] offers a proactive defense against insider threats, reducing false positives and prioritizing high-risk alerts.

Keywords

Cyber Risk, UEBA, CVSS

1. Introduction: Objectives and Goals

The main goals of this paper are in cyber risk management by implementing advanced behavioral & semantic log analysis techniques to identify insider threats efficiently. The approach of semantic log analysis incorporates information like syslog data, access logs, and event logs, and integrates information such as login patterns to discern risks through behavioral analysis. This solution will help in employing targeted mitigation strategies to tackle insider threats. Minimizing false alerts will also ensure that reliability of the security mechanism is maintained.

The existing solutions for insider threat detection mainly focus on rule-based detection, signature-based IDS, etc. These traditional solutions have a problem of false positives and missed detections. Our solution introduces a novel seman-

tic log analysis approach that combines log data with contextual behavioral information. Psychology-based profile analyses may also be integrated into this framework.

Project Structure

Figure 1 represents the structure of the project. We started with gathering Linux System logs from GitHub. The logs were cleaned and processed by removing null values, downsizing the number of logs to match semantic dataset size, and filtering them further to have the most unique but most important sample logs. These logs were finally ready to be integrated with the semantic data. The semantic and behavioral data generation step included fields like login/logout timestamps, employee designation, resource accessed, etc. This dataset was analyzed, and risk score was associated to each field after careful analysis. Once final risk analysis was done, the integrated data was ready to give insider threat score and generate alerts.

2. Existing Work

We describe some related work in the paragraphs that follow that helped guide our approach.

In [2], *the authors* develop a methodology for detecting potentially malicious insider behavior using virtual machine introspection (VMI). They propose a four-step methodology for the development and validation of malicious insider threat alerting using VMI, utilizing a malicious attacker taxonomy to aid identification of observables for monitoring potential malicious actions. The methodology developed is effective in detecting malicious insider scenarios on Windows guests. Validation using two datasets confirms the effectiveness of the identified observables, providing a practical approach for detecting insider threats using VMI. The proposed methodology offers a practical means of detecting insider threats through VMI, addressing the challenges of characterizing, and detecting malicious insiders in real-world systems.

In [3], the authors introduce BIFROST, a statistical analysis-based insider threat detection system deployable to resource-disadvantaged systems. It aims to baseline network profiles and host activities unique to operational environments, alerting system operators to focus monitoring resources on hosts showing potential characteristics of insider activities. The proof-of-concept implementation of BIFROST achieved best- and worst-case detection rates of approximately 74%. BIFROST represents a practical solution for detecting insider threats through statistical analysis. It provides a starting point for future research and development efforts in insider threat detection, offering potential improvements in detection rates and operational effectiveness.

In [4], the authors proposed Warder, an online insider threat detection system leveraging diverse feature dimensions and hypergraph-based correlation techniques. Warder to create an user's daily profile using neural language processing

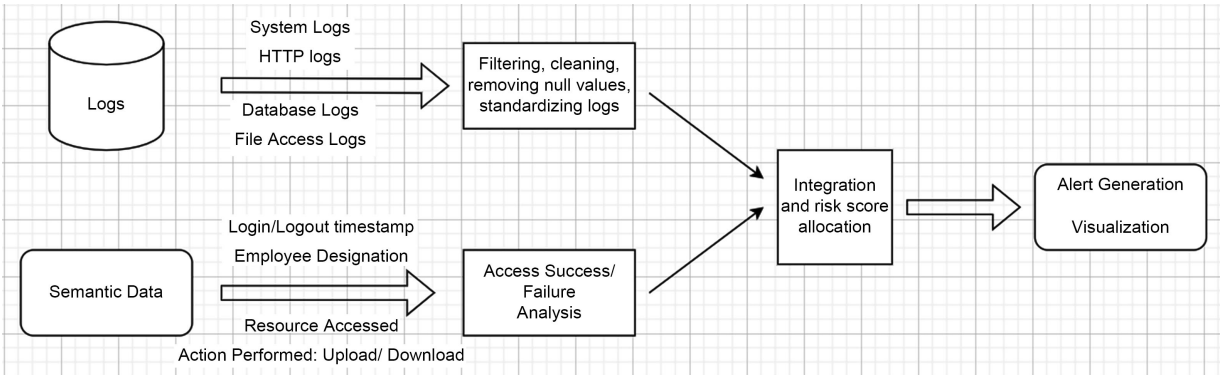


Figure 1. Project structure block diagram.

(NLP) models and correlates suspicious users' activities with threat scenarios to enhance screening effectiveness. Evaluation using the public CMU CERT dataset shows that Warder outperforms competing approaches from the Oxford and CMU groups. This indicates the effectiveness of Warder in detecting insider threat activities compared to existing methods. Therefore, Warder offers a novel approach to online insider threat detection, addressing the limitations of existing methods by incorporating diverse feature dimensions and hypergraph-based correlation techniques. It represents a valuable contribution to the field of insider threat detection, with potential for further refinement and improvement in future research.

3. UEBA—User & Entity Behavior

User and Entity Behavior Analytics (UEBA) plays a pivotal role in modern insider threat detection, particularly when based on log analysis and marks a crucial shift in cybersecurity strategy. UEBA not only encompasses user behavior but also extends its analysis to entity behavior, such as that of cloud applications or unmanaged end-points. This holistic approach enables a more comprehensive understanding of potential threats by correlating the behavior of various entities with user actions. Gartner's introduction of UEBA emphasizes the importance of considering entity behavior alongside user behavior, thereby enhancing threat detection accuracy. By leveraging advanced analytics and modeling, UEBA surpasses the limitations of legacy SIEM systems and traditional UBA. These systems often struggle with false negatives due to lack of context, high maintenance requirements, and alert fatigue from false positives. UEBA addresses these challenges by adopting probabilistic models and risk factors, which replace Boolean alerts and enable more efficient threat detection.

Incorporating UEBA into our work significantly enhanced insider threat detection capabilities. By fusing user and entity behavior analysis with log data, UEBA provides a more nuanced understanding of potential security risks. This approach not only reduces false positives but also enables automated incident response and prioritization of alerts. UEBA's ability to model complex user behavior patterns and detect anomalous activities across diverse log sources aligns

perfectly with the goals of my project, which aims to enrich log data with contextual information for insider threat detection.

Moreover, UEBA's adoption of machine learning techniques further refines threat detection by adjusting risk scores based on peer group comparisons and multiple indicators of anomalous behavior. This dynamic approach to risk assessment goes beyond static threshold-based methods, ensuring more effective detection of sophisticated threats, such as APTs or insider attacks. By incorporating UEBA we can leverage its advanced analytics capabilities to enhance the accuracy and efficiency of insider threat detection, ultimately strengthening the security framework of organizations [5]-[18].

4. Implementation

We now present a detailed description, where the sections describe the steps taken to tie together different aspects of our project.

4.1. Gathering Linux System Logs

Categorization of Log Types: we categorized different types of logs based on their relevance to the project. While logs such as firewall logs may focus on external threats and may not be directly applicable, logs from Linux and Windows servers, file access logs, VPN logs, and database logs were identified as more relevant and useful. These logs provide valuable insights into user activities and system events within the organization's infrastructure.

Relevance of Linux System Logs: Among the categorized log types, Linux system logs emerged as particularly relevant and valuable for insider threat detection. Linux system logs capture a wide range of system activities, including user logins, file accesses, process executions, and system events. These logs provide rich contextual information that is essential for identifying potential insider threats within the organization. These system logs are obtained from GitHub [5]. The Linux logs used in this project are typically stored in the directory /var/log/. Specifically, the dataset was collected from the /var/log/messages file on a Linux server. These logs were gathered over a period of 260+ days as part of the Public Security Log Sharing Site project. /var/log/messages is a common location for system logs on Linux systems. It contains a variety of log entries related to system events, including kernel messages, system startups and shutdowns, user login/logout activities, and various other system-related events. The acquired system logs were voluminous, comprising over 3000 records. However, these records were inconsistent, containing numerous null values and illegible formats. Following extensive research, we opted to down sample the logs to a more manageable 300 records. Subsequently, we undertook a process of cleaning, refining, and standardizing the logs. Once this preprocessing was complete, we imported the logs into Grafana for analysis. During the analysis phase, we identified the most pertinent and crucial Linux logs that would be utilized (**Figure 2**).

	Logs	Description	Impact	Li
2	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*>	Failed authenticatoin attempt via SSH user	4 High	3
3	check pass; user unknown	Failed login attempt for an unknown user	2 Low	3
4	ALERT exited abnormally with [1]	Abnormal exit of a process	4 High	2
5	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=root	Failed authenticatoin attempt via SSH user as root	5 Very High	3
6	Authentication failed from <> (<>): Software caused connection abort	Failed authentication due to software caused abort	3 Moderate	3
7	usbcore: registered new driver hub	Routine registration of new USB driver hub	1 Very Low	5
8	audit: initializing netlink socket (disabled)	Netlink initialization is disabled	2 Low	3
9	ROOT LOGIN ON tty2	Successful root login	5 Very High	1
10	ANONYMOUS FTP LOGIN FROM <*>, (anonymous)	Anonymous FTP login	3 Moderate	2
11	warning: can't get client address: Connection reset by peer	Warning due to a connection reset by the peer preventing retrieval of client address.	2 Low	3
12	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=guest	Authentication failure for the "guest" user via SSH.	3 Moderate	3
13	PCI: Invalid ACPI-PCI IRQ routing table	Error indicating an invalid ACPI-PCI IRQ routing table.	4 High	2
14	Couldn't authenticate user	Authentication failure for user	2 Low	3
15	There is already a security framework initialized, register_security failed.	Failure to register a security framework due to an already initialized framework.	4 High	2
16	Security Scaffold v<*>.<*>.<*> initialized	Security scaffold initialization	1 Very Low	5
17	SELinux: Starting in permissive mode	SELinux starting in permissive mode	2 Low	3
18	Failure registering capabilities with the kernel	Failure to register capabilities with the kernel	3 Moderate	2
19	SELinux: Initializing.	SELinux initialization	1 Very Low	5
20	session opened for user <> by (uid=<>)	User session opened for mentioned user ID	2 Low	4

Figure 2. Clean and standardized linux system logs.

4.2. Linux Log Analysis—Risk Score Allocation

For each log, we calculated a corresponding risk score by quantifying its impact and likelihood as a product of likelihood and impact.

$Risk = (Impact * Likelihood)$

Below are the risk scores and logs that we sampled:

1) uid = 0 euid = 0 tty = NODEVssh ruser = rhost = [host_IP]

• Impact—4 High—The impact is high as it involves a failed authentication attempt with the root user, potentially leading to unauthorized access with elevated privileges.

• Likelihood—3 Moderate—Likelihood is moderate since it involves authentication failure, which may occur occasionally, but still indicates potential security threats.

• Risk score = 12

2) check pass; user unknown

• Impact—2 Low—The impact is low as it involves a failed login attempt for an unknown user, indicating no successful access to the system.

• Likelihood—3 Moderate—Likelihood is moderate since it suggests occasional attempts to access the system with invalid user credentials.

• Risk score = 6

3) ALERT exited abnormally with 1

• Impact—4 High—The impact is high as it signifies the abnormal termination of a critical process or service, potentially leading to system instability or disruption of services.

• Likelihood—2 Low—Likelihood is low since abnormal exits with specific exit codes are relatively rare occurrences.

• Risk score = 8

4) authentication failure; logname = uid = 0 euid = 0 tty = NODEVssh ruser = rhost =[host_IP] user = root

• Impact—5 Very High—The impact is very high as it involves a failed authentication attempt for the root user, which, if successful, would grant full administrative privileges to the intruder.

• Likelihood—3 Moderate—Likelihood is moderate since it involves authentication failure, which may occur occasionally, but still indicates potential security threats.

• Risk score = 15

5) Authentication failed from: Software caused connection abort

• Impact—3 Moderate—The impact is moderate as it indicates a failed authentication due to a connection abort, potentially disrupting the authentication process and causing inconvenience to legitimate users.

• Likelihood—3 Moderate—Likelihood is moderate since software-caused connection aborts may occur occasionally due to various factors such as network issues or system instability.

• Risk score = 9

6) usbcore: registered new driver hub

• Impact—1 Very Low—The impact is very low as it represents a routine system event related to USB driver management, which typically does not affect system functionality or security.

• Likelihood—5 Very High—Likelihood is very high as the registration of new USB driver hubs is a common occurrence during system operation, especially during startup.

• Risk score = 5

7) audit: initializing netlink socket (disabled)

• Impact—2 Low—The impact is low as it is auditing process and disabling it reduces immediate impact.

• Likelihood—3 Moderate—Likelihood is moderate as this event may occur occasionally, especially if auditing features are configured or modified.

• Risk score = 6

8) ROOT LOGIN ON tty2

• Impact—5 Very High—The impact is very high as it indicates a successful login as the root user, granting full administrative privileges to the intruder.

• Likelihood—1 Low—Likelihood is very low since root logins should be rare and tightly controlled, especially directly on a terminal (tty).

• Risk score = 5

9) ANONYMOUS FTP LOGIN FROM [host IP] (anonymous)

• Impact—3 Moderate—The impact is moderate as it indicates an attempt to access the system via FTP using anonymous credentials, potentially allowing unauthorized users to view or modify files.

• Likelihood—2 Low—Likelihood is low as anonymous FTP logins should be restricted, and legitimate use cases for such logins are rare.

• Risk score = 6

10) warning: can't get client address: Connection reset by peer

• Impact—2 Low—The impact is low as it indicates a warning message due to a connection reset by the peer, which may temporarily disrupt communication but does not directly affect system functionality or security.

• Likelihood—3 Moderate—Likelihood is moderate as occasional connection resets by peers may occur due to network issues or communication errors.

• Risk score = 6

11) authentication failure; logname = uid = 0 euid = 0 tty = NODEVssh ruser = rhost = [host IP] user = guest

• Impact—3 Moderate—The impact is moderate as it indicates a failed authentication attempt for the "guest" user, which may lead to unauthorized access with limited privileges.

• Likelihood—3 Moderate—Likelihood is moderate since authentication failures may occur occasionally, especially for commonly used usernames like "guest".

• Risk score = 9

12) PCI: Invalid ACPI-PCI IRQ routing table

• Impact—4 High—The impact is high as it indicates an error in the ACPI-PCI IRQ routing table, which can lead to hardware conflicts, device malfunctions, or system instability.

• Likelihood—2 Low—Likelihood is low as errors in the ACPI-PCI IRQ routing table are relatively rare occurrences.

• Risk score = 8

13) Couldn't authenticate user

• Impact—2 Low—The impact is low as it indicates a failed authentication attempt for an unidentified user, suggesting no successful access to the system.

• Likelihood—3 Moderate—Likelihood is moderate as occasional authentication failures may occur due to various reasons such as mistyped credentials or automated login attempts.

• Risk score = 6

14) There is already a security framework initialized, register-security failed.

• Impact—4 High—The impact is high as it indicates a failure to register a security framework, potentially leading to inadequate security measures and leaving the system vulnerable to security threats.

• Likelihood—2 Low—Likelihood is low since failures to register security frameworks due to already initialized frameworks are relatively rare occurrences.

• Risk score = 8

15) Security Scaffold v[major].[minor].[patch] initialized

• Impact—1 Very Low—The impact is very low as it indicates a routine event related to the initialization of the security scaffold, which is a standard security feature.

• Likelihood—5 Very High—Likelihood is very high as the initialization of the security scaffold typically occurs during system startup or security module loading.

• Risk score = 5

16) SELinux: Starting in permissive mode

• Impact—2 Low—The impact is low as it indicates SELinux starting in per-

missive mode, which allows actions that would normally be denied by SELinux policies but logs them for analysis.

• Likelihood—3 Moderate—Likelihood is moderate as starting SELinux in permissive mode may occur occasionally, especially during troubleshooting or testing.

• Risk score = 6

17) Failure registering capabilities with the kernel

• Impact—3 Moderate—The impact is moderate as it indicates a failure to register capabilities with the kernel, potentially leading to restricted functionality or security vulnerabilities.

• Likelihood—2 Low—Likelihood is low since failures to register capabilities with the kernel are relatively rare occurrences.

• Risk score = 6

18) SELinux: Initializing.

• Impact—1 Very Low—The impact is very low as it indicates a routine event related to SELinux initialization, which is a standard security feature.

• Likelihood—5 Very High—Likelihood is very high as SELinux initialization typically occurs during system startup or security module loading.

• Risk score = 5

19) Session opened for user guest_user by (uid = [uid])

• Impact—2 Low—The impact is low as it indicates a routine event of a user session being opened, which is a standard operation.

• Likelihood—4 High—Likelihood is high as user sessions are frequently opened during system operation.

• Risk score = 8

4.3. Generation of Semantic Dataset

We started by researching examples of semantic data in the context of insider threats. The challenge we encountered was that most available data belonged to organizations with embedded employee details. Further, we did not have access to any organizational data. Therefore, we decided to generate a sample semantic dataset synthetically.

We explored tools such as Mockaroo (see **Figure 3**) and Generate-Data to generate the sample datasets. However, we found that some of the fields provided by these tools were too limited for our project's needs and could not be generated to fit the required dataset. To address this issue, we developed Python scripts (see **Figure 4**). These scripts allowed us to feed custom inputs and generate multiple fields in random order to create a dataset that met my project requirements.

After setting up our Grafana dashboard in Ubuntu VM, we decided to load the semantic dataset into a database (see **Figure 5**). We decided to use PSQL due to its support for complex data types, advanced querying capabilities, scalability, data integrity features, extensibility, and vibrant community. Its ability to handle

structured and semi-structured data efficiently, coupled with robust transactional support and extensibility options, makes it well-suited for storing and querying the diverse types of information found in semantic datasets. Additionally, PostgreSQL's active community and ecosystem ensure ongoing support and development, making it a reliable choice for long-term projects.

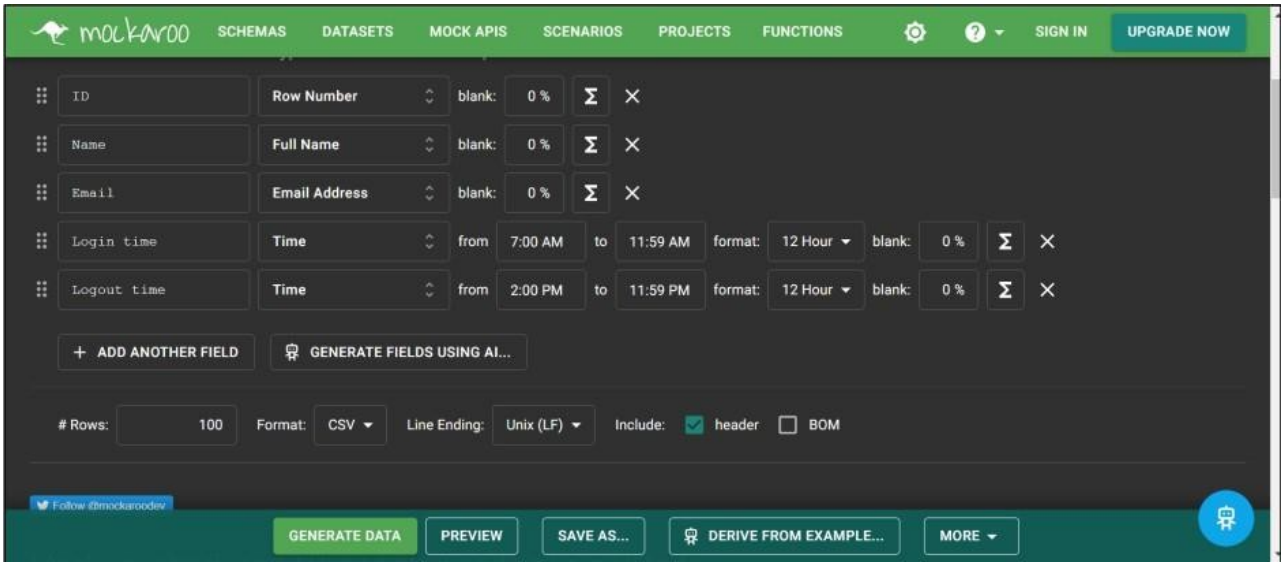


Figure 3. Mockaroo exploration.

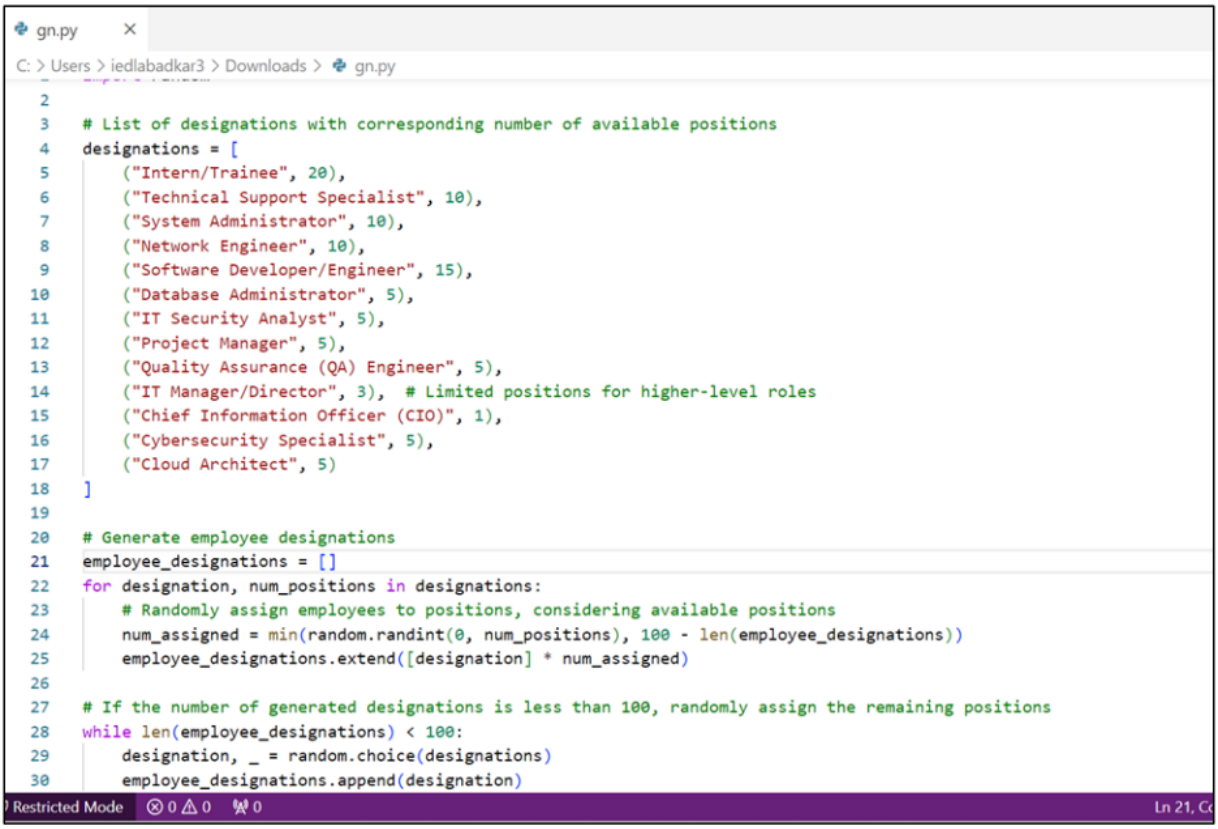
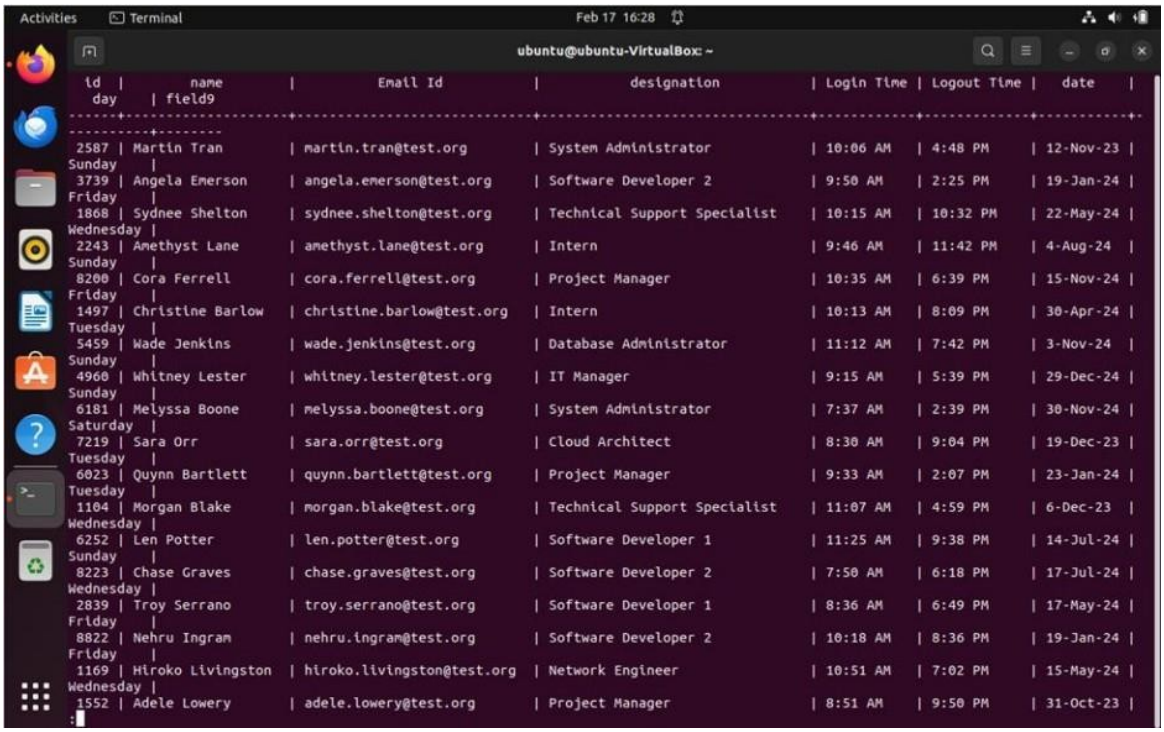


Figure 4. Semantic data script example.



id	name	Email Id	designation	Login Time	Logout Time	date	day	field9
2587	Martin Tran	martin.tran@test.org	System Administrator	10:06 AM	4:48 PM	12-Nov-23	Sunday	
3739	Angela Emerson	angela.emerson@test.org	Software Developer 2	9:50 AM	2:25 PM	19-Jan-24	Friday	
1868	Sydnee Shelton	sydnee.shelton@test.org	Technical Support Specialist	10:15 AM	10:32 PM	22-May-24	Wednesday	
2243	Amethyst Lane	amethyst.lane@test.org	Intern	9:46 AM	11:42 PM	4-Aug-24	Sunday	
8200	Cora Ferrell	cora.ferrell@test.org	Project Manager	10:35 AM	6:39 PM	15-Nov-24	Friday	
1497	Christine Barlow	christine.barlow@test.org	Intern	10:13 AM	8:09 PM	30-Apr-24	Tuesday	
5459	Wade Jenkins	wade.jenkins@test.org	Database Administrator	11:12 AM	7:42 PM	3-Nov-24	Sunday	
4960	Whitney Lester	whitney.lester@test.org	IT Manager	9:15 AM	5:39 PM	29-Dec-24	Sunday	
6181	Melyssa Boone	melyssa.boone@test.org	System Administrator	7:37 AM	2:39 PM	30-Nov-24	Saturday	
7219	Sara Orr	sara.orr@test.org	Cloud Architect	8:30 AM	9:04 PM	19-Dec-23	Tuesday	
6023	Quynn Bartlett	quynn.bartlett@test.org	Project Manager	9:33 AM	2:07 PM	23-Jan-24	Tuesday	
1104	Morgan Blake	morgan.blake@test.org	Technical Support Specialist	11:07 AM	4:59 PM	6-Dec-23	Wednesday	
6252	Len Potter	len.potter@test.org	Software Developer 1	11:25 AM	9:38 PM	14-Jul-24	Sunday	
8223	Chase Graves	chase.graves@test.org	Software Developer 2	7:50 AM	6:18 PM	17-Jul-24	Wednesday	
2839	Troy Serrano	troy.serrano@test.org	Software Developer 1	8:36 AM	6:49 PM	17-May-24	Friday	
8822	Nehru Ingram	nehru.ingram@test.org	Software Developer 2	10:18 AM	8:36 PM	19-Jan-24	Friday	
1169	Hiroko Livingston	hiroko.livingston@test.org	Network Engineer	10:51 AM	7:02 PM	15-May-24	Wednesday	
1552	Adele Lowery	adele.lowery@test.org	Project Manager	8:51 AM	9:50 PM	31-Oct-23		

Figure 5. PSQL Db, semantic data.

4.4. Semantic Dataset—Risk Score Allocation

We began by following the NIST [15] guide and using the CVSS [7] calculator to assign scores to the columns of my semantic dataset. We have included screenshots showcasing the scores of each column, which primarily consist of employees accessing organization resources at specific times of the day. Below figures illustrate the risk scores associated with particular actions taken by employees to access these resources.

We have incorporated actions such as uploading, downloading, and general access for testing purposes. The logs in our dataset mainly capture employee interactions with organization resources, and the risk scores are heavily influenced by CVSS metrics. For instance, the risk score tends to be higher if sensitive resources like log files or databases are downloaded or uploaded, whereas accessing common resources like websites results in lower risk scores.

As illustrated in **Figures 6-8**, the risk scores are categorized by the day of the week, with higher scores on weekends, indicating increased risk when employees access office resources outside regular business hours. Additionally, it also displays risk scores for login and logout timestamps, with higher scores during irregular office hours. Lastly, the dataset includes risk scores based on employee designations, where higher designation levels correspond to higher risk scores associated with the resources accessed.

The final semantic risk score calculation formula is a below where each field mentioned indicates its respective score:

$$(action * resource * day * login * logout)/designation$$

Action	Score
Download	0.9
Upload	0.8
Access	0.2

Resource	Score
Database	8.3
Log File	9.9
Cloud	9.1
Software	6.5
Website	4.2

Semantic Field	Score	Score
Database Access	0.2*8.3	1.66
Database Download	0.9*8.3	7.47
Database Upload	0.8*8.3	6.64
Log File Access	0.2*9.9	1.98
Log File Download	0.9*9.9	8.91
Log File Upload	0.8*9.9	7.92
Cloud Access	0.2*9.1	1.82
Cloud Download	0.9*9.1	8.19
Cloud Upload	0.8*9.1	7.28
Enterprise Software Access	0.2*6.5	1.3
Enterprise Software Download	0.9*6.5	5.85
Enterprise Software Upload	0.8*6.5	5.2
Website Access	0.2*4.2	0.84
Website Download	0.9*4.2	3.87
Website Upload	0.8*4.2	3.36

Figure 6. Action and resource semantic scores.

Designation	Score
System Administrator	0.1
Software Developer 2	0.5
Technical Support Specialist	0.4
Intern	0.5
Project Manager	0.3
IT Security Analyst	0.2
Database Administrator	0.2
IT Manager	0.4
Cybersecurity Engineer	0.3
Cloud Architect	0.5
Software Developer 1	0.5
Network Engineer	0.3
Quality Assurance (QA) Engineer	0.5
Human Resources	0.5

Day	Score
Monday	0.14
Tuesday	0.14
Wednesday	0.14
Thursday	0.14
Friday	0.14
Saturday	0.28
Sunday	0.28

Figure 7. Day and designation semantic scores.

Login	Score	Logout	Score
12:01 am - 3 am	9.9	12:01 am - 3 am	9.9
3:01 am -6 am	8.7	3:01 am -6 am	9.1
6:01 am - 9 am	3.4	6:01 am - 9 am	8.7
9:01 am- 12 noon	2.2	9:01 am- 12 noon	5.6
12:01 pm- 3 pm	5.6	12:01 pm- 3 pm	2.2
3:01 pm - 6 pm	6.7	3:01 pm - 6 pm	3.4
6:01 pm- 9 pm	7.4	6:01 pm- 9 pm	6.7
9:01 pm - 12 midnight	9.1	9:01 pm - 12 midnight	7.4

Figure 8. Login and logout timestamps with scores.

4.5. Python Code and PSQL Database

We chose Python as the programming language to generate the insider threat scores for its modernity and ease of use. Connecting to a PostgreSQL (PSQL) server was straightforward with Python3, making it a suitable choice for our project. Our software architecture consisted of several parts. Firstly, we defined the ranges for semantic scores in a dictionary format, allowing for easy reference and manipulation of semantic data within the code (see Figure 9). We then stored the semantic dataset in a PSQL database, where the code takes the em-

ployee name as user input and retrieves the corresponding row for that employee (see Figure 10). Upon retrieving the employee's record, our code calculates the Insider Threat score by fetching a random Linux log for risk score calculation. If the employee record exists, the code proceeds to calculate the final Insider Threat score as the sum of the Linux log score and the semantic data score (see Figure 11). We ensured that my code handles cases of incorrect inputs and missing records gracefully, implementing appropriate error handling mechanisms. For the Linux log score calculation, a random log is selected from the dataset, and the risk score for this log is determined based on its unique identifier.

By following these steps, we have been able to effectively utilize Python to connect to the PSQL database, retrieve the necessary data, and calculate insider threat scores based on a combination of semantic and Linux system logs.

$$FinalInsiderThreatScore = SemanticScore + LinuxScore$$

```
semantic_scores = {
    "action": {"Download" : 0.9, "Upload": 0.8, "Access": 0.2},
    "resource": {"Database" : 8.3, "Log file": 9.9, "Cloud": 9.1, "Enterprise Software": 6.5, "Website": 4.2 },
    "day": {"Monday": 0.14, "Tuesday": 0.14, "Wednesday": 0.14, "Thursday": 0.14, "Friday": 0.14, "Saturday": 0.28, "Sunday":
0.28},
    "designation": {"System Administrator": 0.1, "Software Developer 1": 0.5,"Software Developer 2": 0.5, "Technical Support
Specialist": 0.4,
    "Intern": 0.5, "Project Manager": 0.3, "IT Security Analyst": 0.2, "Database Administrator": 0.2, "IT Manager": 0.4,
"Cybersecurity Engineer": 0.3,
    "Cloud Architect": 0.5, "Network Engineer": 0.3, "Quality Assurance (QA) Engineer": 0.5, "Human Resources": 0.5 },
    "login": {"12:01 am - 2:59 am": 9.9, "3:00 am - 5:59 am": 8.7, "6:0 am - 8:59 am": 3.4, "9:00 am - 11:59 am": 2.2,
              "12:00 pm - 2:59 pm": 5.6, "3:00 pm - 5:59 pm": 6.7, "6:00 pm - 8:59 pm": 7.4, "9:00 pm - 11:59 pm": 9.1},
    "logout": {"12:01 am - 2:59 am": 9.9, "3:00 am - 5:59 am": 9.1, "6:00 am - 8:59 am": 8.7, "9:00 am - 11:59 am": 5.6,
               "12:00 pm - 2:59 pm": 2.2, "3:00 pm - 5:59 pm": 3.4, "6:00 pm - 9:59 pm": 6.7, "9:00 pm - 11:59 pm": 7.4},
}
```

```
linux_scores = {
    "1": {"description": "authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*>","likelihood": 4, "impact":
3},
    "2": {"description": "check pass user unknown","likelihood": 2, "impact": 3},
    "3": {"description": "ALERT exited abnormally with [1]","likelihood": 4, "impact": 2},
    "4": {"description": "authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=root","likelihood": 5,
"impact": 3},
    "5": {"description": "Authentication failed from <> (<>): Software caused connection abort","likelihood": 3, "impact": 3},
    "6": {"description": "usbcore: registered new driver hub","likelihood": 1, "impact": 5},
    "7": {"description": "audit: initializing netlink socket (disabled)","likelihood": 2, "impact": 3},
    "8": {"description": "ROOT LOGIN ON tty2","likelihood": 5, "impact": 1},
    "9": {"description": "ANONYMOUS FTP LOGIN FROM <*>, (anonymous)","likelihood": 3, "impact": 2},
    "10": {"description": "warning: can't get client address: Connection reset by peer","likelihood": 2, "impact": 3},
    "11": {"description": "authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*> user=guest","likelihood":
3, "impact": 3},
    "12": {"description": "PCI: Invalid ACPI-PCI IRQ routing table","likelihood": 4, "impact": 2},
    "13": {"description": "Couldn't authenticate user","likelihood": 2, "impact": 3},
    "14": {"description": "There is already a security framework initialized, register_security failed.","likelihood": 4,
"impact": 2},
    "15": {"description": "Security Scaffold v<*>.<*>.<*> initialized","likelihood": 1, "impact": 5},
    "16": {"description": "SELinux:  Starting in permissive mode","likelihood": 2, "impact": 3},
    "17": {"description": "Failure registering capabilities with the kernel", "likelihood": 3, "impact": 2},
    "18": {"description": "SELinux:  Initializing.","likelihood": 1, "impact": 5},
    "19": {"description": "session opened for user <> by (uid=<>)","likelihood": 2, "impact": 4},
}
# Display menu for linux logs with numbers and descriptions
```

Figure 9. Dictionary of semantic data and linux logs.

```
#creation of cursor object
cur=conn.cursor()

#employee name input
employee_name=input("\nEnter the full name of the employee you want to calculate risk score for:")

# Execute a SQL query to fetch the row for the given employee name
cur.execute("SELECT action, resource, day, designation, login, logout FROM sticdataset WHERE name = %s", (employee_name,))
row = cur.fetchone()

#fetch all data
cur.execute("SELECT * FROM sticdataset WHERE name = %s", (employee_name,))
record = cur.fetchone()

if record:
```

```
#connect to PSQL dataset
conn=psycopg2.connect(
    dbname="test1",
    user="postgres",
    password="[redacted]",
    host="localhost",
    port= "5432"
)
```

Figure 10. PSQL connection via python code and queries.

```
# Define the formula to calculate the final score
def calculate_score(inputs):
    global semanticscore
    try:
        action_score = semantic_scores["action"][inputs["action"]]
        resource_score = semantic_scores["resource"][inputs["resource"]]
        day_score = semantic_scores["day"][inputs["day"]]
        designation_score = semantic_scores["designation"][inputs["designation"]]
        login_time = inputs["login"]
        logout_time= inputs["logout"]

        login_score = find_score(login_time, semantic_scores["login"])
        logout_score = find_score(logout_time, semantic_scores["logout"])

        print("\naction score", action_score)
        print("resource score", resource_score)
        print("day score", day_score)
        print("designation score", designation_score)
        print("loginscore", login_score)
        print("logout score", logout_score)
        final_score = (action_score * resource_score * day_score * login_score * logout_score) / designation_score
        return final_score
        semanticscore=final_score
    except KeyError as e:
        print(f"invalid input: {e}")
        return None
```

Figure 11. Function to calculate final semantic score.

5. Flowchart

The program starts by prompting the user to enter the name of an employee. It then connects to a PostgreSQL database and retrieves relevant data for the specified employee, including their actions, resources, login/logout times, and other details. The program calculates a semantic score based on this data using predefined scoring dictionaries and formulas. After displaying the semantic score to the user, the program presents a menu of Linux logs and randomly selects one. It calculates a score for this selected log based on predefined likelihood and impact values. The program then combines the semantic score and the Linux log score to compute the final insider threat risk score, which it displays to the user before ending.

User Input: Prompt the user to enter the name of the employee for whom the risk score needs to be calculated.

Database Query: Connect to the PostgreSQL database and execute a query to fetch the relevant data for the specified employee name (action, resource, day, designation, login, logout).

Data Fetching: Fetch the data from the database and store it in variables.

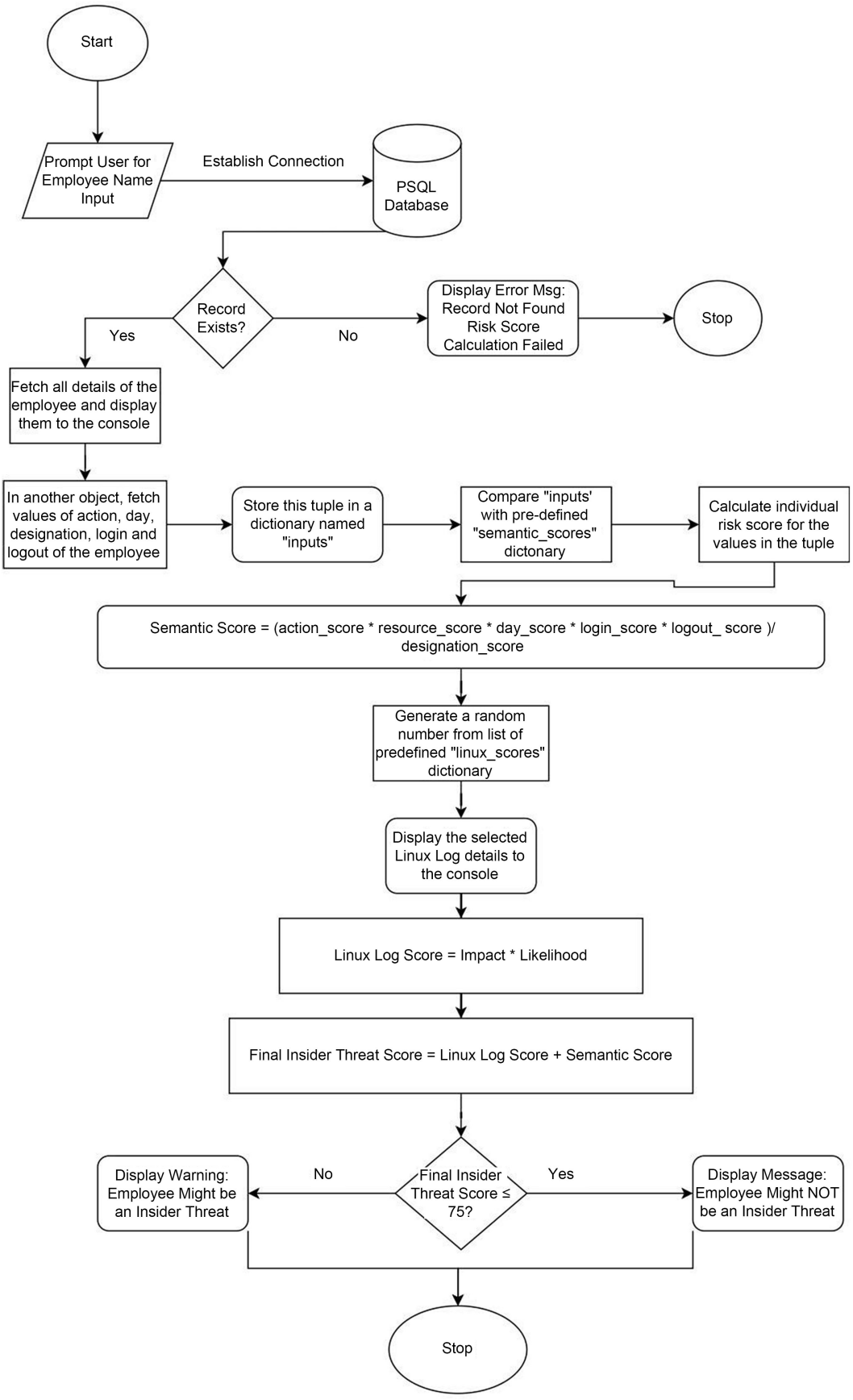
Display Employee Data: If employee data is found, display it to the user.

Calculate and Display Semantic Score.

Select Random Log: Randomly select a log from the menu. Calculate and Display Linux log Score.

Calculate Final Score: Add the semantic score and the Linux log score to get the final insider threat risk score.

Display Final Score: Display the final insider threat risk score to the user. End.


Start
Prompt User for Employee Name Input
Establish Connection
PSQL Database
Record Exists?
Yes
No
Display Error Msg: Record Not Found Risk Score Calculation Failed
Stop
Fetch all details of the employee and display them to the console
In another object, fetch values of action, day, designation, login and logout of the employee
Store this tuple in a dictionary named "inputs"
Compare "inputs' with pre-defined "semantic_scores" dictonary
Calculate individual risk score for the values in the tuple
Semantic Score = (action_score * resource_score * day_score * login_score * logout_ score)/ designation_score
Generate a random number from list of predefined "linux_scores" dictionary
Display the selected Linux Log details to the console
Linux Log Score = Impact * Likelihood
Final Insider Threat Score = Linux Log Score + Semantic Score
Final Insider Threat Score ≤ 75?
No
Yes
Display Warning: Employee Might be an Insider Threat
Display Message: Employee Might NOT be an Insider Threat
Stop

6. Results

The Screenshots of the Test Cases are below. These represent the output after code execution.

6.1. Test Case 1

The employee is identified as an insider threat.

Figure 12 represents the output of the code when the employee is identified as an insider threat. As shown, the code takes employee name as the input and displays the employee record. It is illustrated that the employee is a system admin (higher privilege) and downloads a sensitive logfile outside office hours on weekend. As per the semantic risk score formula, the score is high. Additionally, random Linux log is assumed to be generated for the given employee and final insider threat score is shown.

6.2. Test Case 2

The employee is not identified as an insider threat.

Figure 13 represents the output of the code when the employee is not identified as an insider threat. As shown, the code takes employee name as the input and displays the employee record. It is illustrated that the employee is an technical support specialist (lower privilege) and accesses websites during office hours on a weekday. As per the semantic risk score formula, the score is low. Additionally, random Linux log with ID = 17 is assumed to be generated for the given employee and final insider threat score is shown.

6.3. Test Case 3

The employee record does not exist.

Figure 14 represents the output of the code when the employee record does not exist. As shown, the code takes employee name as the input and throws the error when the record fetching fails. As per the Flowchart (Figure 14), the Linux score is not even calculated because the execution failed at previous level of finding the employee record.

6.4. Test Case 4

Random Linux log allocation for the same employee.

Figure 15 represents the output of the code when the employee is not identified as an insider threat with demonstration of random Linux log selection. As shown, the code takes employee name as the input and displays the employee record. It is illustrated that the employee is a technical support specialist (lower privilege) and accesses websites during office hours on a weekday. As per the semantic risk score formula, the score is low. Additionally, random Linux log with ID = 3 is assumed to be generated for the same employee and final insider threat score is shown.

7. Comparison with Existing Work

7.1. Detection Methodology

Existing solutions primarily rely on rule-based detection and signature-based intrusion detection systems (IDS). These methods are often expensive to implement, rely on historical data, and may not provide real-time detection. Furthermore, they tend to be rigid and inflexible in their detection approach. In contrast, our solution offers an innovative insider threat detection scheme that enriches log data with contextual information. This approach enables real-time detection instead of periodic scanning, resulting in more timely identification of potential threats. Additionally, the flexibility and customization inherent in our solution allow organizations to tailor the detection process to their unique needs and vulnerabilities.

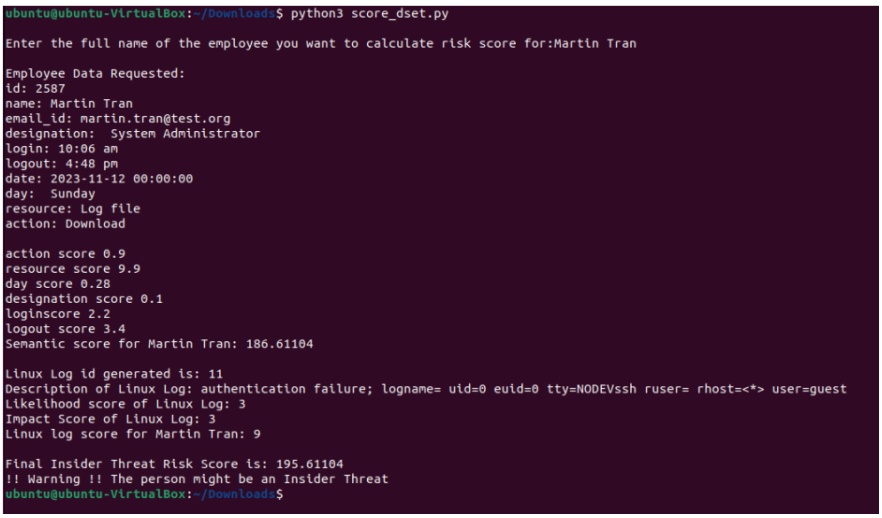


Figure 12. Insider threat.

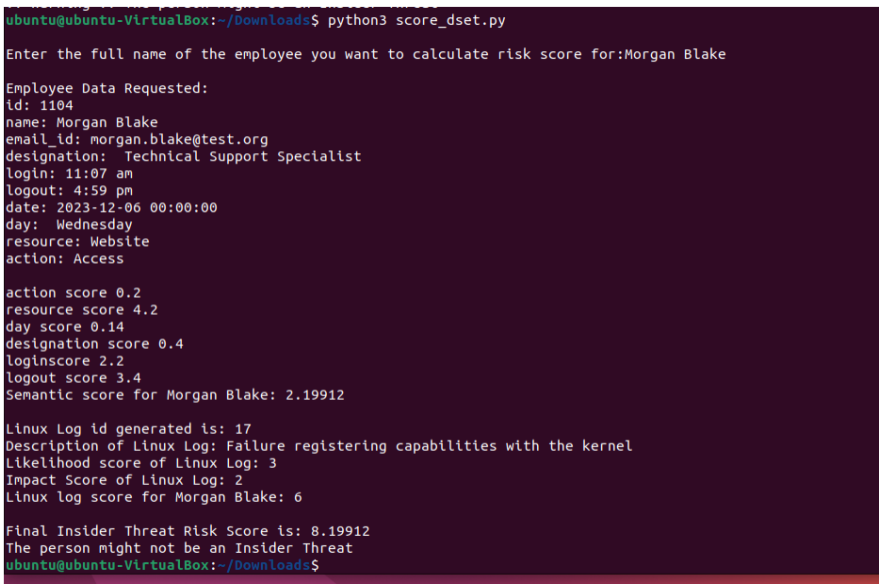


Figure 13. Not insider threat.

```
ubuntu@ubuntu-VirtualBox:~/Downloads$ python3 score_dset.py

Enter the full name of the employee you want to calculate risk score for:Irawati Edlabadkar
No data found for employee: Irawati Edlabadkar
Error occured during record search

!! Final Insider Threat Risk Score calculation FAILED !!
ubuntu@ubuntu-VirtualBox:~/Downloads$
```

Figure 14. No record.

```
ubuntu@ubuntu-VirtualBox:~/Downloads$ python3 score_dset.py

Enter the full name of the employee you want to calculate risk score for:Morgan Blake

Employee Data Requested:
id: 1104
name: Morgan Blake
email_id: morgan.blake@test.org
designation:  Technical Support Specialist
login: 11:07 am
logout: 4:59 pm
date: 2023-12-06 00:00:00
day:  Wednesday
resource: Website
action: Access

action score 0.2
resource score 4.2
day score 0.14
designation score 0.4
loginscore 2.2
logout score 3.4
Semantic score for Morgan Blake: 2.19912

Linux Log id generated is: 3
Description of Linux Log: ALERT exited abnormally with [1]
Likelihood score of Linux Log: 4
Impact Score of Linux Log: 2
Linux log score for Morgan Blake: 8

Final Insider Threat Risk Score is: 10.19912
The person might not be an Insider Threat
ubuntu@ubuntu-VirtualBox:~/Downloads$
```

Figure 15. Random log selection.

7.2. False Positive Rates

Rule-based and signature-based detection methods are known to suffer from high false positive rates, as they compare observed activities with known patterns and rules. This can lead to unnecessary alerts and increased operational overheads. Our solution mitigates this issue by adopting a simple approach to log analysis that focuses on enriching log data with semantic information. By doing so, our solution reduces the occurrence of false positives, leading to more accurate and actionable alerts for insider threats. We have also added future work possibilities using ML to enhance false positive rate mitigation.

7.3. Scalability and Cost Effectiveness

Implementing and maintaining traditional insider threat detection systems can be costly and resource-intensive, particularly when additional data collection processes or computations are required. Our solution leverages existing Linux logs generated automatically as part of routine employee activities. This eliminates the need for additional data collection processes or computations, making the detection process more scalable and efficient. Additionally, the cost effectiveness of my approach reduces the financial burden associated with implementing and maintaining insider threat detection systems.

7.4. Adaptability to Emerging Threats

Traditional rule-based and signature-based detection methods may struggle to adapt to emerging or novel insider threat tactics. These methods are often limited by their reliance on predefined rules or patterns. In contrast, our solution offers a more adaptive approach to insider threat detection. By enriching log data with contextual information and leveraging real-time detection techniques, our solution can more effectively identify and respond to evolving insider threat tactics, even those that may not have been previously encountered.

7.5. Integration with Organizational Policies

Existing solutions may lack the ability to seamlessly integrate with organizational policies and procedures, leading to gaps in coverage or conflicts with existing security measures. Our solution addresses this challenge by offering flexibility and customization options. By allowing organizations to tailor the detection process to their unique needs and vulnerabilities, our solution can better align with organizational policies and procedures, ensuring a more cohesive and comprehensive approach to insider threat detection.

7.6. Ease of Deployment and Management

Implementing and managing traditional insider threat detection systems can be complex and time-consuming, requiring significant expertise and resources. Our solution aims to simplify the deployment and management process by offering a user-friendly interface and streamlined configuration options. By providing a simple and intuitive approach to log analysis, our solution reduces the burden on IT teams and facilitates more efficient management of insider threat detection efforts.

8. Problems Faced

8.1. Absence of Real-World Data

One of the primary challenges we faced was the absence of real-world data and reliance on sample data for log analysis techniques. Sample data often fails to accurately reflect the complexity of actual activity and threats encountered in real-world scenarios. To address this challenge, we explored various approaches to simulate realistic but synthetic data scenarios and validate the system's performance under different conditions.

8.2. Volume of Logs

Managing the volume of logs proved to be another significant challenge. Preprocessing logs is necessary before conducting analysis due to the sheer volume of data generated by systems and applications. To manage this challenge effectively, we opted to focus on a subset of logs, prioritizing critical sources such as syslogs and event logs. This enabled me to streamline the analysis process and allocate resources more efficiently.

8.3. Lack of Semantic Data

Another challenge we encountered was the lack of readily available semantic data for analysis. Semantic data, which provides deeper insights into the meaning and context of log entries, is typically embedded within organizational data, making it challenging to access and utilize effectively. To address this challenge, we explored techniques for extracting and incorporating semantic data into the analysis framework, leveraging existing tools and methodologies where possible.

8.4. Variation in Risk Score Metrics

Defining and calculating risk scores presented challenges due to variation across different frameworks and standards. The inconsistency in risk score metrics made it difficult to maintain consistency and comparability. To address this issue, we decided to integrate the risk matrix with established frameworks such as the National Institute of Standards and Technology (NIST) and the International Organization for Standardization (ISO) 27001. Additionally, we referred to resources such as the CERT guide to insider threats to ensure alignment with industry best practices and guidelines.

In conclusion, while these challenges posed significant hurdles during the development and implementation of the insider threat detection system, careful consideration and strategic planning enabled me to overcome them effectively. By addressing these challenges head-on, we enhanced the robustness and reliability of the system, ultimately improving its effectiveness in detecting insider threats.

9. Future Work

9.1. False Positives Mitigation

To address the issue of false positives in insider threat detection, following strategies would make a good solution. Refinement of Analysis Algorithms: This approach involves continuous refinement of analysis algorithms to minimize false positives. This entails fine-tuning algorithms to enhance their ability to differentiate between genuine threats and benign activities. This method would leverage machine learning techniques to enable continuous learning and adaptation of the system. By incorporating machine learning, our system could dynamically adjust its detection capabilities based on evolving patterns and behaviors, thus reducing false positives over time. Providing additional context and supporting evidence for alerts is essential for mitigating false positives. Incorporating contextual information such as user behavior patterns and system configurations would validate alerts and provide a clearer understanding of potential threats.

9.2. Integration of Diverse Log Sources

The project could integrate logs from diverse systems and applications to enhance the effectiveness of insider threat detection. In addition to Linux logs, we could integrate logs from various systems and applications, including Windows,

network devices, and cloud services. This comprehensive approach would enable us to gain a holistic view of the organization's IT environment and improve detection accuracy across different platforms. By integrating logs from diverse sources, our system can detect insider threats more accurately, even across different platforms. This capability is crucial for detecting sophisticated attacks that may involve multiple systems or services.

9.3. User-Friendly Interface

Ensuring ease of use and adoption is essential for the success of our insider threat detection system. To achieve this, the future goal would be focusing on designing a user-friendly interface with the following features. The system could have an intuitive dashboard that allows users to easily navigate logs, drill down into specific events, and take appropriate actions in response to detected threats. We can provide customizable dashboards and reporting features to enhance usability. This allows users to tailor the interface to their specific needs and preferences, ultimately improving their experience with the system. By prioritizing user-friendly design principles, aim to enhance usability and facilitate efficient monitoring and response to insider threats.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Exabeam (2023) What Is UEBA and Why It Should Be an Essential Part of Your Incident Response. https://www.exabeam.com/explainers/ueba/what-is-ueba-and-why-it-should-be-an-essential-part-of-your-incident-response/

[2] Crawford, M. and Peterson, G. (2013) Insider Threat Detection Using Virtual Machine Introspection. 2013 46*th Hawaii International Conference on System Sciences*, Wailea, HI, 7-10 January 2013, 1821-1830. https://doi.org/10.1109/HICSS.2013.278 https://ieeexplore.ieee.org/abstract/document/6480061

[3] Findley, S., Singh, G., Shaffer, A., *et al.* (2019) A Statistical Analysis Framework for Detecting Insider Threat Activities on Cyber Systems. 2019 *International Conference on Computational Science and Computational Intelligence* (CSCI), Las Vegas, NV, 5-7 December 2019, 1-6. https://doi.org/10.1109/CSCI49370.2019.00008 https://ieeexplore.ieee.org/abstract/document/9071044

[4] Jiang, J., Chen, J., Gu, T., *et al.* (2019) Warder: Online Insider Threat Detection System Using Multi-Feature Modeling and Graph-Based Correlation. 2019 *IEEE Military Communications Conference* (*MILCOM*), Norfolk, VA, 12-14 November 2019, 1-6. https://doi.org/10.1109/MILCOM47813.2019.9020931 https://www.researchgate.net/publication/339760983

[5] GitHub Link to Linux Logs, GitHub Repository. https://github.com/logpai/loghub?tab=readme-ov-file

[6] Software Engineering Institute (2022) Common Sense Guide to Mitigating Insider Threats, Seventh Edition.

https://insights.sei.cmu.edu/library/common-sense-guide-to-mitigating-insider-threats-seventh-edition/

[7] National Vulnerability Database. https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

[8] BeyondTrust (2022) Insider Threat Indicators: How to Identify & Mitigate Insider Attacks. https://www.beyondtrust.com/blog/entry/insider-threat-indicators-how-to-identify-mitigate-insider-attacks

[9] Insider Threat Guide (2024) Insider Threat: The Ultimate Guide. https://www.nextdlp.com/resources/blog/insider-threat-ultimate-guide

[10] (2022) ISO/IEC 27001:2022: Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. https://www.iso.org/standard/27001

[11] Ekran System (2021) Portrait of Malicious Insiders: Types, Characteristics, and Indicators. https://www.ekransystem.com/en/blog/portrait-malicious-insiders

[12] Axelsson, S. (2000) The Base-Rate Fallacy and the Difficulty of Intrusion Detection, *ACM Transactions on Information and Systems Security*, **3**, 186-205. https://doi.org/10.1145/357830.357849

[13] Cappelli, D.M., Moore, A.P. and Trzeciak, R.F. (2012) The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes, Addison-Wesley Professional, 2012.

[14] Medium (2021) How to Use Grafana for Data Visualization. https://medium.com/nightingale/how-to-use-grafana-fordata-visualization-39d62276fcf9

[15] National Institute of Standards and Technology (NIST) (2015) Specifications, Tolerances, and Other Technical Requirements for Weighing and Measuring Devices. https://www.nist.gov/system/files/documents/2017/04/28/hb44-15-web-final.pdf

[16] Raut, M., Dhavale, S., Singh, A. and Mehra, A. (2020) Insider Threat Detection Using Deep Learning: A Review. 2020 3*rd International Conference on Intelligent Sustainable Systems* (*ICISS*), Thoothukudi, 3-5 December 2020, 856-863. https://doi.org/10.1109/ICISS49785.2020.9315932 https://ieeexplore.ieee.org/abstract/document/9315932

[17] Macak, M., Vanát, I., Merjavý, M., Jevočin, T. and Buhnova, B. (2020) Towards Process Mining Utilization in Insider Threat Detection from Audit Logs. 2020 *Seventh International Conference on Social Networks Analysis, Management and Security* (*SNAMS*), Paris, 14-16 December 2020, 1-6. https://doi.org/10.1109/SNAMS52053.2020.9336573 https://ieeexplore.ieee.org/abstract/document/9336573

[18] Vasudevan, P. (2021) Detect Insider Threats with User Behavior Analytics. https://www.ibm.com/blogs/digital-transformation/in-en/blog/detect-insider-threats-with-user-behavior-analytics/