Proactive Cloud Threat Hunting Through Adversary Emulation

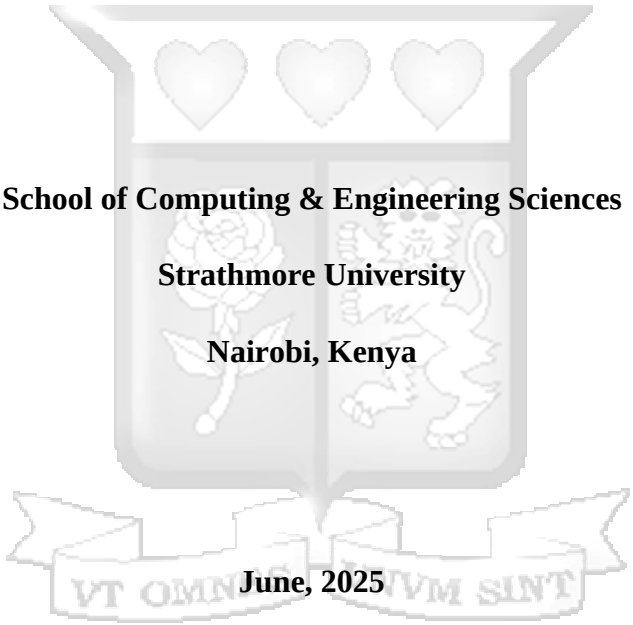
by

Bunde, Collins Oduor

136999

Submitted in partial fulfillment of the requirements for the Master of Science in

Information Systems Security at Strathmore University



School of Computing & Engineering Sciences

Strathmore University

Nairobi, Kenya

June, 2025

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student's Name: Collins Oduor Bunde

Sign:  Date: 26/05/2025

Approval

The dissertation of Collins Oduor Bunde was reviewed and approved for examination by the following:

School of Computing & Engineering Sciences,
Strathmore University

Dr. Ondřej Ryšavý
Lecturer, School of Computing & Engineering Sciences
Faculty (iLab)

Dr. Julius Butime,
Dean, School of Computing & Engineering Sciences,
Strathmore University

Prof. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Abstract

Organizations have increasingly adopted cloud computing infrastructure as the foundation for delivering digital services, a trend known as 'digital transformation.' While cloud computing offers flexibility, cost reduction, and improved productivity, there are also significant security concerns due to reduced visibility and an increased attack surface. The emergence of cloud-conscious threat actors exacerbates these concerns. This research focused on operationalizing cloud threat hunting as a proactive measure to reduce attacker dwell time in organizational environments. The aim was to review threat-hunting approaches for the cloud environment, concentrating on threats targeting IAM misconfigurations. This included analyzing adversary emulation methods that can support threat-hunting in the cloud, developing and testing a threat-hunting model for operationalization, and validating the performance of the threat model. The study's objectives were achieved through a design science approach, employing an experimental methodology divided into offensive and counter-offensive phases. For the offensive phase, adversary emulation provided a comprehensive summary of common threat scenarios. For the counter-offensive phase, three hypotheses were formulated based on MITRE ATT&CK techniques: Hypothesis 1 focused on T1078.004, Hypothesis 2 on T1098.003, and Hypothesis 3 on T1136.003. The study demonstrated the effectiveness of the developed threat-hunting model in identifying cloud-specific threats. All three hypotheses, which focused on key IAM misconfiguration attack vectors, were validated as true, highlighting the importance of proactive threat hunting for these attack vectors.

Keywords: Cloud Threat Hunting, IAM Misconfigurations, Adversary Emulation, MITRE ATT&CK Techniques

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Amazon EC2	Amazon Elastic Compute Cloud.
API	Application Programming Interface
AWS	Amazon Web Services
CI/CD	Continuous Integration and Continuous Delivery/Deployment
GCP	Google Cloud Platform.
IaaS	Infrastructure-As-a-Service
IAM	Identity and Access Management
MITRE ATT&CK	A Knowledge base of Adversarial Tactics, Techniques, and Common Knowledge
OEM	Original equipment manufacturers
PIR	Priority Intelligence Requirement
SIEM	Security Information and Event Management
SOC	Security Operation Center
TDIR	Threat Detection and Incidence Response
TH	Threat hunting
TTPs	Tactics, Techniques, Procedures

Definition of Key Terms

PIR (Priority Intelligence Requirement):	A specific intelligence requirement that emphasizes critical information vital to an organization, frequently applied in the context of security and threat analysis. (First, 2025).
TDIR (Threat Detection and Incident Response):	The processes and methodologies employed to identify, analyse, and respond to security threats and incidents within an organization. (Lenaerts-Bergmans, 2025)
TH (Threat Hunting)	The proactive search for signs of malicious activities or threats in an IT environment aimed at identifying and mitigating security vulnerabilities before they can be exploited. (Daszczyszak et al., 2019).
TTP's	The behaviours and methodologies of cyber adversaries, detailing how they operate in specific contexts to achieve their objectives during an attack. (Alexis Ahmed, 2023)

Acknowledgements

As I reflect on the journey of pursuing my Master's dissertation, I can't help but feel incredibly grateful to Strathmore University for the wonderful opportunity to study for a Master of Science in Information Systems Security (MSc ISS). I truly appreciate the support from all my colleagues and lecturers who shared their ideas and insights, guiding me to bring this dissertation to fruition. A special thanks goes out to Dr. Oziany for his thoughtful assistance during my search for a supervisor while I was relocating from Kenya. I am particularly indebted to my supervisor, Dr. Ondřej Ryšavý, whose unwavering guidance has been invaluable throughout this process. His insightful contributions have been instrumental in shaping this work, and I sincerely appreciate all his help

Chapter 1: Introduction

1.1. Background of the Study

Threat hunting (TH) is an iterative process to search out, identify, and understand adversaries within a specific organization's network. According to a survey conducted by SANS, threat hunting is arguably one of the growing areas where most organizations channel their investments (Lee & Lee, 2018). Threat hunting primarily focuses on threats but also heavily depends on threat intelligence. A threat actor should have these three characteristics: intent, capability, and opportunity to cause harm (Daszczyszak et al., 2019). The new focus on threat hunting emphasizes the initiatives and efforts of analysts, who have purposefully set out to identify and counteract adversaries already within an organization's network.

Cloud security comprises a set of procedures and technologies designed to address both internal and external threats to business workloads in a cloud environment. As organizations undergo digital transformation, they must enhance the security of their assets by incorporating cloud-based security tools. Recently, cloud computing infrastructure has become the backbone of the delivery pipeline for every digital service. Gartner, a research and survey firm, predicts that global spending on cloud services will reach $679 billion in 2024 (Howley, 2023). The cloud is becoming indispensable, with companies migrating significantly from on-premises infrastructure to the cloud. However, securing assets, particularly in the cloud, has proven challenging due to a lack of proper understanding of the shared responsibility model among original equipment manufacturers (OEMs) such as AWS, GCP, and Microsoft Azure and their clients.

Hence, organizations have changed their focus towards strengthening their cyber defenses by investing in new capabilities and fundamentally reimagining how they think about security in the cloud. One key to this was operationalizing cloud threat hunting to reduce the dwell time of potential adversaries within a given cloud environment.

In cybersecurity, adversary emulation refers to imitating or simulating a threat actor's tactics, techniques, and procedures (TTPs) to assess how effectively an organization's defenses function (Alexis Ahmed, 2023). This study aims to conduct threat hunting in the AWS cloud through adversary emulation, replicating attacker TTPs as discrete or atomic tests designed to evaluate security defenses within an environment. Attack scenarios are created and executed against a target environment to help identify security gaps and validate controls.

Operationalizing cloud threat hunting in this research involves setting objectives, gathering intelligence, assessing controls, developing scenarios, using tools, monitoring for detections, and measuring performance for ongoing improvement. This structured approach can enhance security, detect threats proactively, and mitigate risks in cloud environments.

One way this research attempts to improve security for the cloud environment is by operationalizing threat hunting to help identify threat actors in any of these cloud environments, riding on the attacker's tactics, techniques, and procedures. Of importance is the fact that conventional approaches, such as signature-based and anomaly-based detection, are not adaptable to the ever-changing cloud threat landscape.

According to the SANS survey by Lee & Lee (2018), the adversary's overall dwell time within an organization's network averaged about 90 days. This means the overall impact of a cyber breach might be pretty high when an adversary is not detected early enough. Similarly, the level of compromise in itself might be quite costly for any business to recover. The results of such cyber intrusions include financial loss, data breaches, and data exposures.

Conventional adversary detection and defense techniques operate loosely connected activities (Araujo et al., 2021). They might not be effective in the current situation with the emergence of cloud-conscious threat actors like COZY BEAR, COSMIC WOLF, or SCATTERED SPIDER. These Nation-state and criminal adversaries are using cloud infrastructure to host phishing lure documents and malware as well as implementing command-and-control channels on top of existing cloud services (CrowdStrike, 2023)

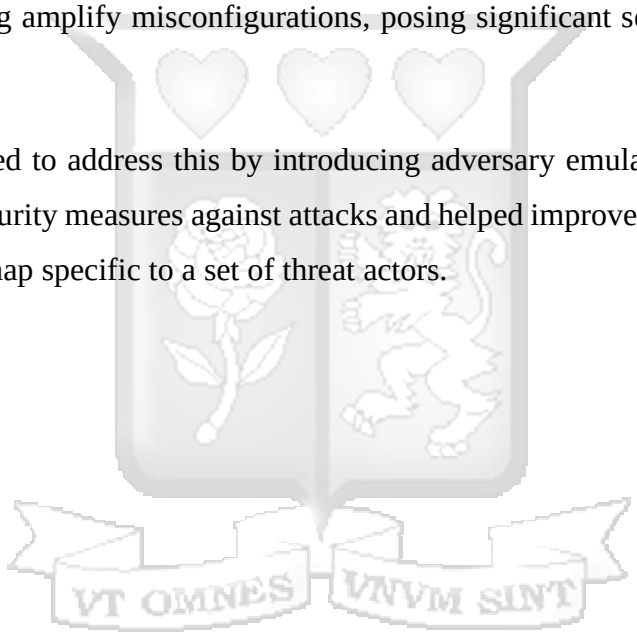
According to CrowdStrike's (2023) cloud security risk report, identity has become the key cloud access point, with at least 43% of the cloud intrusion observed by CrowdStrike. This proves that current attacks are targeted and designed to exploit the cloud's unique characteristics. Therefore, while endeavoring to capture the dimension features of cloud persistent threats, defense strategies should do more in developing awareness of the more extensive attack campaign, especially with the advent of Initial access brokers (IAB), whose leading role in the attack continuum is to gain and sell access. Cloud threat hunting enables security organizations to respond to single out-of-context events and develop a clear picture of the attack landscape.

AWS's current threat-hunting techniques encompass log analysis using CloudTrail to monitor API calls and user activities, along with anomaly detection powered by machine learning to

spot deviations from standard behavior. IAM user monitoring tracks account creation and modifications. Malicious activities can be identified through network traffic analysis using VPC Flow Logs and Amazon GuardDuty. At the same time, real-time alerts are generated via automated threat detection with GuardDuty and AWS Security Hub. Key strategies involve creating incident response playbooks, integrating threat intelligence feeds, leveraging user behavior analytics (UBA), And monitoring particular API requests linked to malicious activities.

On the other hand, these techniques face challenges because the shared responsibility model limits visibility and leaves organizations blind to malicious attacks, distributed cloud services in multi-cloud infrastructures and hybrid environments expand the attack surface, and rapid automation and scaling amplify misconfigurations, posing significant security threats for the cloud environment.

This research attempted to address this by introducing adversary emulation, which provided realistic testing for security measures against attacks and helped improve detection capabilities by riding on a threat map specific to a set of threat actors.



1.2 Problem Statement

Cloud environments, particularly AWS, face an expanding attack surface and an evolving threat landscape that demand adaptive security strategies. However, traditional threat detection mechanisms often lack proactive methodologies for identifying sophisticated cloud threats. This study proposed an innovative approach to operationalizing threat hunting with adversary emulation in the cloud as a suitable strategy for detecting adversary behavior within any organization's cloud workloads. The research explored how adversary emulation can be used to support threat hunting efforts as an effective strategy to enhance detection capabilities, with a concentration on IAM threats in AWS.

1.3 Objectives

1.3.1 General Objectives

The study aimed to operationalize cloud threat hunting with adversary emulation as an effective cyber defense strategy in detecting cloud threat actors with a focus on IAM misconfiguration in AWS. By solving this research problem, organizations improve their cloud security posture and protect themselves from imminent attacks.

1.3.2 Specific Objectives

a. To review threat-hunting approaches for the cloud environment.

b. To analyze adversary emulation methods that can be used to achieve threat hunting in the cloud

c. To develop and test a platform for operationalizing threat hunting through adversary emulation.

d. To validate the performance of the threat-hunting platform.

1.4 Main Research Question.

a. How can adversary emulation be leveraged to operationalize cloud threat hunting as an effective cyber defense strategy for detecting IAM threats in an AWS environment?

1.4.1 Research Question

a. What methods are employed for threat hunting in cloud environments?

b. What methods of adversary emulation can be utilized for threat hunting in the cloud?

c. How can we develop and test a platform for operationalizing threat hunting that enhances detection in the cloud?
d. How effective is the threat-hunting platform at identifying threats?

1.5 Justification

Organizations are increasingly using cloud computing for digital services, a trend termed 'digital transformation.' While it offers flexibility and cost savings, cloud computing also raises significant security concerns due to lower visibility and a larger attack surface. Operationalizing cyber threat hunting in the cloud involves a proactive 'assume breach' philosophy instead of relying solely on endpoint detection tools to find potential threats (Microsoft Security Experts, 2022). Although the latter method combats automated and known attacks, today's human-operated attacks are more complex. Proactive threat hunting identifies environmental weaknesses and uncovers hidden security threats.

The dynamic cloud computing environment makes conventional threat-hunting methods insufficient for detecting threats. Adversary emulations can enhance the effectiveness of cloud threat hunting, helping to identify novel and emerging threats overlooked by traditional techniques. The study emphasized the necessity of adversary emulation for improving detection and defense in cloud-adopting organizations, including banks and fintech firms.

1.6 Scope and Limitations of the Study

This research focused on threat hunting in AWS cloud infrastructure and attack emulation. It evaluated the cloud enterprise environment and initiated threat-hunting operations by hypothesizing threat actor behaviors. One limitation was the constrained scope of attacker techniques, as the vastness and emergence of cloud threats posed challenges. The study did not fully analyze all attack vectors, concentrating instead on Identity and Access Management (IAM) threats in AWS.

It acknowledged the dynamic cloud threat landscape and emphasized the necessity of objective threat intelligence sources, relying only on vendor-sourced security reports, which may be biased. Independent security reports are recommended for objectivity. Furthermore, the tools for emulating attacks had limited capabilities regarding TTPs and their accuracy, meant to demonstrate the value of this approach for cloud threat hunting.

1.7 Dissertation Organization

The remaining of the dissertation is organized into the following five chapters:

i. Chapter 2: Literature Review – Reviews related work in cloud threat hunting, adversary emulation, IAM misconfigurations, and the use of threat intelligence.
ii. Chapter 3: Methodology – Outlines the methodological approach to meet the research objectives.
iii. Chapter 4: Design – Presents the design approach on which the actualization of the implementations is hinged on.
iv. Chapter 5: Implementation and Findings – Summarizes the implementation of the counter-offensive and offensive components of the research.
v. Chapter 6 Discussion – Discusses how the objectives were achieved, providing recommendations and outlining future work.

1.8 Conclusion

This introduction addresses the need for advanced security strategies in cloud environments by operationalizing threat hunting through adversary emulation, focusing on AWS and IAM threats. It shows that traditional detection methods fail against modern threat actors who exploit cloud infrastructure's unique features. Adversary emulation simulates real-world attacker behaviors, providing a proactive approach to identifying security gaps and enhancing detection capabilities.

The objectives were met by reviewing threat-hunting approaches, analyzing adversary emulation methods, and developing a platform to operationalize threat hunting, validating its effectiveness against IAM threats. While limited to AWS and specific attack vectors, it highlights the value of an “assume breach” mindset and leveraging threat intelligence to counter sophisticated adversaries.

This work underscores the importance of continuous adaptation in cloud security practices, recommending further research into independent threat intelligence sources and broader attack scenarios. The findings lay a foundation for proactively defending cloud assets against a dynamic threat landscape.

Chapter 2: Literature Review

2.1 Introduction

This chapter presented the literature review and introduced the theoretical framework that underpinned the study's theory. It explored key thematic areas, including threat hunting in the AWS Cloud, cyber threat intelligence, adversary emulation, and the MITRE ATT&CK framework for the cloud, while identifying gaps that were addressed in the research.

2.2 Cloud Threat Hunting

Threat hunting is crucial because, at its core, it helps identify an organization's most valuable assets (core elements) and the potential motives attackers might have for targeting them. This is achieved by analyzing business strategy and leadership communication, referred to as Priority Intelligence Requirements (RedHat-infosec, 2022). This forms the foundation of a robust threat intelligence process, aligning enterprise essence with security needs and setting the stage for mapping adversarial threat operations.

Despite the increasing threats to cloud infrastructure, every cloud provider maintains security best practices outlined in their “Well-Architected Framework" to protect their customers from cloud-specific risks (Chris et al., 2022). It is also important to note that many customers remain inherently vulnerable due to the cloud's common security design issues: misconfigurations, insufficient cloud architecture and strategy, inadequate identity and key management, and insecure interfaces. Consequently, these pervasive design issues form the foundation for the initial threat assessment necessary before developing a hunt. Shahar (2022) further emphasizes that standard cloud threat modeling during the design phase of cloud architecture creates a foundation for effective threat hunting. Therefore, this perspective does not diminish the importance of threat modeling as an essential element of the entire threat-hunting journey, as it allows teams to foresee a cyber-attack chain and implement preventive measures before an attack occurs.

A fully operational AWS infrastructure requires the threat-hunting team to assess the environment for threats and undetected breaches. Effective threat hunting involves collecting, analyzing, and visualizing data. According to Chris, Binil, and Abbas (2020), key data sources for threat hunting in AWS include AWS Config, VPC Flow Logs, Amazon CloudWatch, and AWS CloudTrail. Regularly examining logs from these sources can reveal unusual trends and

information about potential dangers. In the context of cloud threat hunting, threat hunters need expertise in analyzing cloud API logs, which are often overlooked as a security threat (Craig, 2022). Understanding the nuances of these logs and attacker tactics (cloud TTPs) is essential for proactive defense, given that attackers increasingly exploit cloud vulnerabilities according to security reports (CrowdStrike, 2023).

From a security control perspective, identity access management (IAM) services are paramount to any cloud tenant, and proper configuration is, therefore, essential for the adequate security of the cloud environment. IAM allows any cloud customer to manage AWS users, groups, and roles, with each entity having assigned permissions that control their level of privileged actions, thereby preventing unauthorized access to protected resources (Sperotto Steen & Khasuntsev, 2021). However, it is important to note that errors often occur during the configuration of these IAM systems, which may lead to potential compromise (Sperotto Steen & Khasuntsev, 2021). Common misconfigurations include attaching IAM policies to users instead of groups, having users with excessive permissions, not deleting older privileged policy versions that may lead to IAM policy rollback attacks, and having resource-based policies not attached to the required resource. The existence of such misconfigurations can serve as a foundation for threat cloud hunting.

Different researchers have devised different approaches to threat hunting. However, most of these approaches often require manual query construction and overlook the valuable external knowledge provided by automating these processes. Such manual and labor-intensive processes can be error-prone and time-consuming.

Gao, Peng, et al. (2021) proposed an evidence-based knowledge system using unsupervised learning to facilitate cyber threat hunting in computer systems to address these challenges. The system leverages system auditing frameworks and incorporates an unsupervised, lightweight, and accurate Natural Language Processing (NLP) pipeline to extract structured threat behaviors from unstructured ASCII text (Gao, Peng et al., 2021). Khorshed et al. (2012) used the support vector machine technique from a statistical machine learning theory and identified the top attacks with an accuracy of 97.13%. Whereas Wang et al. (2012) also proposed an approach that provided an effective means to reconstruct the attack profiles and evaluative countermeasures in the evolutionary process of security management in the cloud. Although

these researchers focus on log-based cyber threat hunting using open-source Cyber Threat Intelligence, they do not specifically discuss conducting threat hunting in the cloud.

Hillier and Karroubi (2022) argue that using AI in threat hunting is critical and has great potential to broaden the scope and increase threat hunting automation to assist cybersecurity professionals. To support this reasoning, Nour et al. (2023) point out the new trend in adopting ML/AI techniques in threat hunting, adding that it has become a focal point of research in recent years, given that it helps automate the threat-hunting process.

Conversely, Ajmal et al. (2021) highlight the limitations of reactive approaches to cybersecurity and the advantages of proactive strategies, such as threat hunting through adversary emulation, in strengthening organizational defenses. They also propose a novel hybrid model for identifying tactics, techniques, and procedures (TTPs) via offensive security, particularly threat hunting supported by adversary emulation. Similarly, Chris et al. (2022) concur with Ajmal et al. (2021) on the need for more proactive measures in enhancing organizational cloud security defenses by promoting better preparedness through asset inventory and Identity and Access Management and staying updated on current threat intelligence.

In a recent review, McCloskey et al. (2023) examined AWS CloudTrail attacks in the wild. They outline key objectives for threat hunting by advocating for identifying breaches and malicious activities, which can inform the development of new detection rules, enhance subject matter expertise on threat activity, and facilitate the documentation of findings for broader knowledge sharing. Their research highlights the importance of investigating AWS CloudTrail logs, which consolidate user activity and API usage across an organization's AWS infrastructure, to detect malicious actions proactively. They also provide actionable queries and indicators of compromise to assist organizations in identifying signs of the observed threat activity in their environments (McCloskey et al., 2023).

Chen et al. (2022) identify several challenges that must be addressed to create effective threat models, including weak signal detection, imbalanced datasets, a lack of high-quality labels, and insufficient clear storylines. While their research covers offensive and counter-offensive aspects, they do not discuss threat hunting in cloud environments, revealing a gap that requires further examination. They also thoroughly analyze threat models using various algorithms for

efficacy. These techniques include the Local Outlier Factor (LOF), Isolation Forest (iForest), and Density-Based Spatial Clustering of Applications with Noise (DBScan). LOF identifies outliers based on their lower density compared to neighboring data points, while the Isolation Forest algorithm isolates outliers through random feature selection and partitioning. On the other hand, DBScan effectively groups points based on distance measurements and clusters data with similar densities. This comprehensive analysis emphasizes the importance of employing various approaches to enhance cybersecurity threat detection capabilities.

Given threat actors' diversity and motivations, a standard threat-hunting process starts with the premise that attackers are present in an environment. To identify these adversaries, a five-phase approach, illustrated in Figure 1 below, is employed.

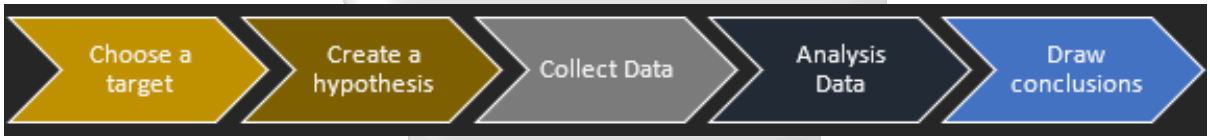


Figure 2.1:Threat Hunting Process

To conduct threat hunting, the hunter chooses a target, formulates a testable hypothesis for valuable results, collects data, analyses the collected data, and draws conclusions.

In summary, a misconfiguration in the AWS cloud, stemming from cloud services like IAM, would form an excellent basis for curating hunting expeditions. The collective insights also highlight the critical need for proactive threat hunting and integrating advanced technologies, such as machine learning, to enhance cloud security measures. Additionally, most researchers have enumerated the importance of an offensive and counter-offensive approach riding on AI/ML, but none has done it with a focus on the cloud infrastructure.

2.3 Cyber Threat Intelligence

Cyber threat intelligence provides valuable information about threat actors, such as threat actor TTPs, which threat hunters can use to develop analytics and hunt for malicious activities matching such TTPs. Most cyber-aware organizations allocate a cyber risk reserve fund to cover cyber-attack costs by investing in cyber threat intelligence. Cyber threat intelligence is crucial in enhancing an organization's resilience to cyber-attacks and may also reduce the overall cost of the cyber reserve fund.

Nour et al. (2023) explain how a data source is a key element in the hunting process and the design of a threat-hunting architecture. This data source can be either internal, from endpoint devices, or external, which refers to data outside the organization. In this case, CTI is categorized as an external data source, which security experts must validate before its use in the hunting process. Various researchers have relied on CTI as a trusted data source, enabling them to make faster, informed decisions to enhance their security defenses and operationalize threat hunting. While existing literature emphasizes the significance of CTI in improving security posture, a notable gap exists regarding its integration with specific threat detection algorithms in cloud environments.

2.4 Mitre's ATT&CK Framework for Cloud

The MITRE ATT&CK framework is described by Nour et al. (2023) as a de facto standard in documenting adversary behaviors and facilitating the discovery and detection of new threats. MITRE, the non-profit organization behind the framework, has envisioned expanding coverage to the cloud environments. Currently, the MITRE ATT&CK Enterprise Matrix's subset, the Cloud Matrix, maps particular tactics, techniques, and procedures (TTPs) that threat actors might employ in attacks against cloud systems. It includes Office 365, Google Workspace, Azure AD, SaaS, and IaaS platforms. Some of the primary use cases include Understanding the adversary's TTPs to guide security policy implementation, identifying gaps in deployed security products or tools, and assessing how effective the security strategy is from a comprehensive perspective.

Building upon the insights the MITRE ATT&CK framework provides, developing a practical threat-hunting roadmap in AWS requires focusing on key areas related to AWS services, configurations, logs, and security controls. This strategic approach not only aids in understanding the AWS environment but also helps pinpoint potential attack surfaces. Table 2.1 outlines specific threat actor behaviors pertinent to this research, mapping notable adversary activities to the MITRE ATT&CK framework and highlighting the impacted AWS services alongside the actual event activities that could be generated. Defenders can better understand how threat actors operate by aligning threat-hunting strategies with the ATT&CK framework, enhancing data visibility, and strengthening their security posture.

Table 2.1: AWS investigation guide, focusing on important API calls adopted from Expel

MITRE Att&ck Tactic	AWS service	IAM Resource	API Calls
Initial Access	Console/CLI		Console Login GetFederationToken GetSessionToken StartSession GetAuthorizationToken
Persistence	AWS IAM		CreateUser CreateRole UpdateAssumeRolePolicy CreateAccessKey
	Amazon EC2		CreateInstance CreateKeyPair CreateImage CreateRepository PutImage
Privilege Escalation	AWS IAM	User	CreateUser CreateLoginProfile UpdateProfile PutUserPolicy AttachUserPolicy AddUserToGroup
		Role	CreateRole AssumeRole CreateInstanceProfile UpdateAssumeRolePolicy AttachRolePolicy PutRolePolicy
		Policy	CreatePolicy CreatePolicyVersion SetPolicyVersion

Despite the widespread use of MITRE ATT&CK, its cloud matric subset is still not widely used, and its application in cloud security remains underexplored. This indicates a need for research that leverages the framework to enhance threat detection models, threat hunting, and incidence response.

2.5 Adversary Emulation in the Cloud

In cyber security, adversary emulation refers to imitating or simulating the tactics, methods, and procedures (TTPs) of a threat actor or adversary to evaluate how well an organization's defenses work (Alexis Ahmed, 2023). Adversary emulation can be instrumental in operationalizing threat hunting by providing a realistic and comprehensive set of attack techniques that can be used to simulate real-world adversary behavior in enterprise environments.

Recent research highlights the rise of automated threat emulation, with tools like Lore and Infection Monkey gaining popularity for their automation capabilities (Holm & Sommestad, 2024). The Atomic Red Team and the Metasploit framework are commonly used tools during cyber defense exercises. On the other hand, Sheikhi and Kostakos (2023) developed a tool to simulate adversary group behaviors, generating realistic threat data to evaluate threat models. Meanwhile, Bakker (2022) utilized MITRE Caldera as a platform for autonomous adversary emulation, demonstrating its effectiveness in enhancing cybersecurity preparedness.

Figure 2.2:The Adversary Emulation Life Cycle

Although recent studies emphasize its significance, little is known about the potential applications of adversary emulation in cloud environments and how it can work with machine learning methods to enhance detection. This integration could significantly improve the

accuracy and effectiveness of threat detection systems. Several key approaches have been identified to achieve effective adversary emulation in the cloud. Realistic emulation, detection engineering, and scenario-based testing are essential for effective cloud threat hunting.

i. Realistic emulation leverages threat actor techniques to generate actionable insights for cyber defenders. This enables threat hunters to simulate adversary behavior and understand how attackers operate within specific cloud environments.

ii. Detection engineering employs technologies like Stratus Red Team across platforms like Kubernetes, AWS, GCP, and Azure. This approach allows organizations to assess and enhance their detection and response capabilities, identify gaps, and refine detection rules to better identify malicious activities.

iii. Scenario-based testing facilitates the execution of attack tactics within cloud environments. It offers threat hunters practical scenarios to evaluate their ability to detect and respond to shared threats in cloud systems, improving overall security postures.

In conclusion, further research is needed to explore the effective integration of adversary emulation with machine learning algorithms in cloud environments to enhance the accuracy and efficiency of threat detection systems. Tools like Stratus Red Team, capable of mimicking attacks on major cloud platforms such as AWS, GCP, and Azure (Tafani, 2022), can significantly contribute to these research efforts.

2.5.1 Adversary Emulation Techniques

By adopting prospective attackers (TTPs) in their emulation endeavors, businesses can detect vulnerabilities and flaws that may not be discernible through regular security assessments. According to Cyberranges (2023), adversary emulation techniques typically revolve around these three areas:

1. Understanding APT groups and their operations - The first step is to profile the specific APT groups of interest by understanding the threat actor's name, their motivations if known, and recent attacks and targets. This includes assessing their level of sophistication, previous operations, TTPs, and the areas they target.
2. Analyzing APT groups and their operations — The MITRE Navigator is useful for effectively mapping their TTPs across the attack chain to analyze the threat group.

Additional frameworks like the MITRE ATT&CK framework and the Diamond Model help to gain a deeper understanding of the threat actor.

3. Developing an Adversary Emulation Plan — The final stage is creating an emulation plan. This is where a decision is made on the specific threat actor to be investigated. Fundamentally, three primary questions should be answered in this phase: Do they target my industry? What is their geographical location? Is there sufficient CTI or organization-outsourced CTI? Once a decision is made on a threat actor, the emulation techniques are identified; as a best practice, the emulation plan should include all post-compromise techniques. Finally, understand the attack chain followed by the threat actor. The output in this phase should be a robust, searchable emulation plan (Cyberranges, 2023).

From the above, we can create a test library for recurring emulations that is updated occasionally. McCloskey et al. (2023), in their review of notable attack paths in the AWS cloud trail, provide a comprehensive summary of threat scenarios, such as privilege escalation, that can be assessed during the threat-hunting process. Alternatively, detection opportunities can be designed for these scenarios to help uncover instances of possible compromise in the AWS environment. The Rhino Security Labs article on privilege escalation in AWS also enumerates key IAM privilege escalation attack vectors (Gietzen, 2021).

For instance, an actor with the "*iam:CreatePolicyVersion*" permission may create a new version of a policy to which they have access, allowing them to set their custom permissions, such as designating the policy as the default. This results in the ability to escalate their privileges and gain full Administrator access to the AWS account.

Similarly, an actor with the "*iam: SetDefaultPolicyVersion*" permission can escalate their privileges by utilizing existing policy versions that are not in use, especially if they are not set as default. The impact depends on the policy's permission level, ranging from no privileges to full administrator access, highly dependent on inactive policies.

Additionally, an actor with the "*iam: CreateAccessKey*" permission may create access keys for different users in the AWS environment using the command "aws iam create-access-key --user-name target_user." The potential impact is that the attacker obtains the same permission levels as the target user.

Finally, an actor with the "*iam:CreateLoginProfile*" permission can generate a password that allows a user without a login profile to access the AWS console. In this case, the threat actor may have the same permissions as the user, including administrator access.

These insights provide a way to develop a threat library of tests that can be conducted for continued review of privileged management policies during threat hunting, augmenting the process of developing a comprehensive emulation. plan

2.6 Gaps in Existing Study

Threat hunting in AWS cloud environments requires a deep understanding of the unique vulnerabilities that come with the cloud, especially the critical issue of misconfigurations. This review highlights that IAM misconfigurations are particularly significant and frequently exploited by threat actors. As organizations steadily embrace the cloud, ensuring the security of IAM has become essential for maintaining a strong cloud security posture.

Adversary emulation has become a powerful technique for validating the effectiveness of security controls, but its application in cloud environments is still underexplored. While tools like Stratus Red Team are making strides in this area, there is still a notable lack of research focused on cloud-centric threat emulation. At the same time, the growing enthusiasm for leveraging AI and machine learning in automating threat detection has mostly centred on traditional on-premise systems, with only limited adaptation to cloud-native contexts.

A thorough review of the literature across these areas reveals a clear research gap: the lack of a unified threat hunting strategy within AWS that brings together IAM misconfiguration detection, adversary emulation, and the practical use of cyber threat intelligence (CTI). Currently, existing practices often treat these elements separately, missing out on potential synergies that could greatly enhance proactive threat detection.

To fill this gap, this study suggests creating an integrated threat hunting model for AWS. By combining adversary emulation with CTI and a focused evaluation of IAM vulnerabilities, this research aims to develop a practical, proactive framework for effectively detecting and responding to threats in cloud environments.

2.7 Proposed Conceptual Framework

Based on the analysis of the thematic areas and identified gaps, the proposed conceptual framework integrates Cyber Threat Intelligence (CTI) and threat emulation. The MITRE ATT&CK Framework employs a threat-hunting model utilizing an ML algorithm for anomaly detection. This approach addresses the need for proactive security measures in cloud infrastructure, offering a promising path forward in cloud security.

It requires prioritizing security controls based on critical assets. The framework leverages threat intelligence to ensure the organization's security is updated with the latest threat intelligence, which may be used to develop attack emulation strategies. This attack emulation helps enhance threat hunting and develop new threat detection rules. The framework advocates for implementing attack emulation techniques to identify weaknesses in IAM security controls and encourages ongoing research and development to adapt to the evolving threat landscape. This framework aims to strengthen an organization's overall security posture in the cloud.

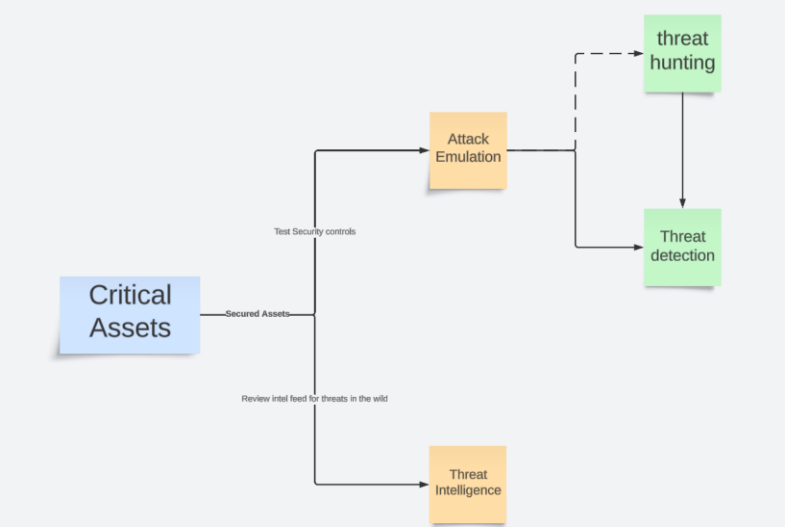


Figure 2.3:The Conceptual Framework

2.8 Conclusion

In summary, this chapter addresses two research questions: how threat hunting can be conducted in the AWS cloud with a focus on IAM and how threat emulation can be implemented in the cloud. The study highlights gaps in each of the key thematic areas, emphasizing the need to develop a novel threat-hunting approach that employs both offensive tactics—developing an adversary emulation plan that may include creating a test library—and counter-offensive aspects—entailing the creation of a threat-hunting model that leverages AI/ML and utilizes CTI to guide the hunting efforts.

Chapter 3: Methodology

3.1 Introduction

Based on the literature review in Chapter 2, which explored threat hunting in the cloud context and adversary emulation methods for enhanced threat detection in the cloud, this chapter covers the methodology used to implement the threat-hunting platform. It illustrates a systematic and procedural approach to achieving these objectives.

The methodology is a strategic framework for this research, directing the journey from foundational assumptions to research design and data gathering (Myers, 2009). This chapter describes the research design, system development methodology, and data collection methods used in the study. This approach stems from the conceptual framework derived from the identified gaps in the literature review. Its goal is to demonstrate how CTI and threat emulation work together to enhance the threat-hunting platform.

3.2 Research Methodology

A common question that arises during research is the choice of research method, which can either be qualitative or quantitative (Goundar, 2012). This research uses a design science approach. It employs an experimental methodology that focuses on how machine learning techniques can be applied to detect threats in the AWS cloud environment. The methodology is divided into:

3.2.1 Theoretical Foundation

This methodology is based on a thorough literature review regarding adversary emulation and threat hunting within the Cloud. Both automated and manual testing experiments were performed within an AWS environment, employing a machine learning model for detecting anomalies in the threat model. This method arises from the conceptual framework shaped by the gaps identified in the literature review. The aim is to illustrate how CTI and threat emulation collaborate to improve the threat-hunting platform.

3.2.2 Research Approach

The study employs a design science approach suitable for developing the threat-hunting platform. While design science is often used to create and evaluate artifacts to find solutions to problems, this research adopts it as a proof of value in the design and implementation of the TH platform.

The design science research (DSR) approach is appropriate for this study as it emphasizes innovative, practical solutions to real-world problems, specifically detecting sophisticated threats in AWS cloud environments. This research aims to design and evaluate a threat hunting platform that uses adversary emulation to identify vulnerabilities and threat actors, aligning with DSR's problem-centered nature. The iterative cycles of artifact creation and refinement in DSR allow for continuous platform improvement based on iterative evaluation tests and threat hunting findings, ensuring the solution is both effective and theory-based.

Moreover, DSR integrates theoretical knowledge—such as cyber threat intelligence, adversary tactics, and machine learning—with practical application in cloud security. This approach supports addressing evolving threats and adapting to changing cloud architectures, making it ideal for threat hunting in AWS. Overall, design science provides a rigorous yet flexible framework that ensures the research delivers academic contributions and a practical, validated artifact to enhance cloud defense capabilities.

3.3 Threat Hunting Model

According to Nour et al. (2023), threat-hunting investigations typically follow three methodologies: proactive investigation, investigation based on Indicators of Compromise/Indicators of Attack (IoC/IoA), and analytics-based investigation. Among these, the analytics-based approach stands out due to its incorporation of machine learning driven by advanced analytics. This research adopted the analytics-based methodology and employed a threat-hunting model founded on unsupervised learning. In this approach, events are grouped into clusters based on specified metrics, enabling the detection of hidden patterns critical for identifying unusual activities associated with threat actors.

To achieve this, the research leveraged the Isolation Forest algorithm, a specialized unsupervised learning method designed for anomaly detection. Rooted in the principle of isolation, this algorithm efficiently separated outliers through random feature selection and partitioning (Chen et al., 2022). Unlike other tree-based ensemble methods, such as random forest, the Isolation Forest algorithm required fewer splits—or path lengths—to distinguish anomalies, making it highly effective for outlier detection. Its ability to identify anomalies

without relying on labeled data makes it an ideal choice for this research, which focuses on detecting covert patterns of threat activity.

Figure 3.1: TH funnel analogy

This study utilized the threat-hunting model as a screening instrument as in Figure 3.1, analogous to a funnel process. AWS CloudTrail Events were incorporated into the model and trained to predict anomalies. As the logs progress through the model, it identified and excluded events that deviated from established normal behavior. These anomalies, regarded as potential threats, were recorded in a .csv file (e.g., detected_cloudtrail_anomalies.csv) for subsequent analysis. This funnel methodology facilitates the efficient detection of suspicious activities within the extensive volumes of data derived from CloudTrail events.

3.4 The Threat Hunting Platform Methodology Approach

The study aimed to replicate a traditional enterprise cloud architecture similar to an actual setting. Its goal was to validate the overall concept of threat hunting within a genuine environment. Consequently, the broader aim of threat hunting was to facilitate and assist organizations through a security function. This setup incorporated minimal security measures while ensuring an accurate asset inventory and that security protocols were documented for the evaluations.

Figure 3.2 shows that the research implementation was achieved in two phases, namely;

i. **Offensive phase:** Testing stories were created from extracted TTPs, which were acquired from threat intelligence reports, blogs, research papers, and the MITRE Framework. From this phase, attacks were launched using tools like the Stratus Red

Team, with consideration of existing security controls and known vulnerabilities, and a risk score was created.

ii. **Counteroffensive—**A central log collection to facilitate log analysis was a prerequisite for this second research stage. The threat-hunting process started in this phase, from hypothesis building to validation. The successful accomplishment of these stages was considered a successful hunt.

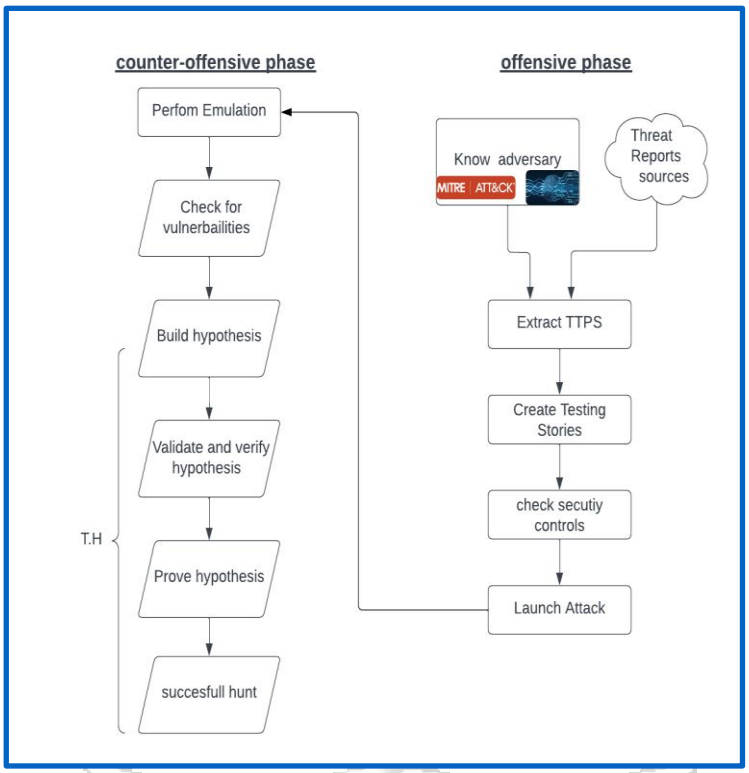


Figure 3.2:Hunting model and Adversary Emulation approach

3.5 Platform Validation and Testing

Each phase encompassed testing and validation, as these elements were crucial for evaluating the efficacy of the threat-hunting platform in detecting threats. Key metrics for this assessment included the number of newly identified occurrences attributable to threat hunting, the implications for security posture, the reduction in dwell time, the quality of detections, and the identification of adversary tactics, techniques, and procedures (TTPs). Splunk (2023) recommends employing the PEAK framework to evaluate the quality of threat-hunting endeavors. The threat-hunting platform's evaluation criteria were based on these established criteria.

i. Upon selection of the attack scenarios, their effectiveness is evaluated during the offensive phase, emphasizing the detectability of each scenario.

ii. Conducting an experimental evaluation of the threat-hunting model involves comparing the results with ground-truth data to calculate precision through cross-validation and identify outliers.

iii. Analyzing the results to discern the strengths and weaknesses of the threat-hunting platform.

3.6 Inclusion and Exclusion Criteria

The inclusion and exclusion criteria were critical to the study's integrity and relevance. The appropriateness and completeness of the research was analyzed in the following ways:

Appropriateness

Alignment with Threat Models: The inclusion criteria specifically focus on adversarial behaviors and tactics relevant to cloud environments. This involves selecting threat actors whose methods align with known cloud vulnerabilities, such as the creation of IAM users, misuse of APIs, and data exfiltration techniques. The research yielded actionable insights by ensuring that the chosen subjects represented real-world threats in cloud settings.

Completeness

Clear Definitions: To effectively evaluate adversaries, the criteria included specific definitions of which adversaries and tactics are part of the study. For example, inclusion parameters in this study are anchored on AWS Cloud IAM misconfigurations.

Demographic and Behavioral Parameters: The types of user behaviors considered for inclusion included privilege escalation and persistence techniques, as well as the frequency of access denied errors or patterns of API calls associated with reconnaissance efforts. This completeness in behavioral parameters allowed for a more nuanced understanding of potential attacker patterns.

3.7 Data Collection

Data collection for the implementation of the threat-hunting platform included:

i. **Cyber Threat Intelligence Reports:** This involves gathering threat intelligence. Most of these reports are collected from cybersecurity solution vendors, while some are open source.

ii. **Public Incident Logs dataset:** Initially, a publicly accessible cloud dataset was acquired as part of the data collection process. Specifically, Scott Piper's AWS CloudTrail JSON dataset, made available for use in AWS security-focused research, was utilized (Piper, 2020). According to Piper (2020), the dataset was designed to instruct defenders in log analysis and the strategies employed by attackers. This research utilized a 2 GB dataset, comprising at least 18 JSON log data files, to train a threat-hunting model that identifies malicious behaviors associated with a specific threat actor. The dataset was sourced from flaws2.cloud and comprises attack-oriented logs related to AWS configurations.

3.8 Ethical Approval

This research prioritized obtaining informed consent, and an ethics clearance certificate, as noted in Appendix B, was awarded before proceeding. All data utilized were thoroughly anonymized to protect individual identities and minimize potential risks. The research safeguards the privacy and confidentiality of participants by ensuring that personal information remains confidential and that participants' identities are protected.

3.8.1 Due Regard for Welfare Rights and Beliefs

This research values and honors each individual's welfare, rights, beliefs, perceptions, customs, and cultural heritage. We ensure that the dataset used is carefully anonymized to safeguard personal identities and reduce any potential risk of privacy leakage. Furthermore, the researcher is committed to setting aside personal biases. By embracing these important aspects, our research establishes a strong foundation of trust and integrity, resulting in positive and ethically sound outcomes.

3.8.2 Fair Distribution and Access Benefits of the Research

The equitable distribution and access to benefits derived from research, particularly in the context of threat hunting in AWS cloud environments, entail several key considerations.

i. **Open Access to Data:** The dataset utilized in this research is publicly accessible, ensuring that the study can be reliably reproduced by others. This openness fosters an environment where researchers, practitioners, and organizations can replicate the work, build on its findings, and adapt methods to suit their unique contexts.

3.9 Conclusion

In summary, these study objectives were achieved through a design science approach, employing an experimental methodology divided into offensive and counter-offensive phases. As part of the counter-offensive phase, an analytics-based threat hunting methodology was used to develop the threat hunting model, utilizing the isolation forest algorithm. Data was collected through publicly accessible CTI reports and a public incident log dataset, which were used to train the model. The next chapter will build on this methodological framework to develop and test the threat hunting platform, demonstrating how these approaches to actualizing the research

Chapter 4: Design

4. 1 Introduction

Chapter 4 introduces the design and evaluation of a cloud threat-hunting platform in AWS, addressing two key objectives: platform development and performance validation. The chapter is divided into two phases: an offensive phase, which uses cyber threat intelligence to emulate attacker behavior based on the MITRE ATT&CK framework, and a counter-offensive phase, which applies a machine learning–based model to detect threats. The workflow includes intelligence collection, TTP mapping, adversary emulation, and anomaly detection in IAM activity logs.

4.2 Offensive Phase: Threat Emulation and Platform Setup

This phase outlines how CTI informed the creation of a threat library and guided adversary emulation within AWS, providing a foundation for testing the platform's detection capabilities using realistic attack scenarios.

4.2.1 Threat Intelligence Collection (CTI-Driven Design Input)

Information about threats targeting the AWS environment was collected to enhance understanding of real-world attack patterns. The objective was to identify distinctive attacker behaviors, validate the current research findings, and develop actionable queries and indicators of compromise. This effort aims to enable more effective threat emulation and detection within the AWS ecosystem.

Threat intelligence for this research was gathered from multiple sources, including blogs, threat reports from cloud security experts, and cloud solutions vendors. These sources were chosen because they are well-respected security providers with proven expertise in threat intelligence and a broad presence in various attack scenarios. Additionally, these vendors are skilled at converting raw threat data into actionable insights, making it more accessible for this research.

Table 4.1 Collected Threat Intelligence

Report source	Technique	Threat Actor
Red Canary Report	**Cloud account:** These accounts serve as gateways to extensive and intricate cloud infrastructure	

	containing critical and sensitive information that's valuable to enterprises and adversaries alike (Red Canary, 2024)	
	API abuse from cloud-specific stealers targeting AWS credentials. Adversaries then leverage these credentials to perform AWS API actions like creating new users, generating access keys, and escalating permissions through built-in IAM roles. (Red Canary, 2024) - *Steals authentication tokens through phishing and abuses those credentials to persist in an environment*	SCATTERED Spider's web –
CrowdStrike report	The report highlights several threat actors that target the cloud. Some notable cloud-conscious adversaries are mentioned in the report (CrowdStrike, 2023)	SCATTERED SPIDER (crime) COZY BEAR *(Russia-nexus)* COSMIC WOLF *(Turkey-nexus)* LABYRINTH CHOLLIMA *(North Korea-nexus)*
Datadog state of cloud security report 2023	**Cloud Accounts:** AWS cloud credentials are challenging to protect since the access keys are static and do not expire. This is regarded as a significant cause of AWS breaches. At least on IAM users do not MFA enabled McCloskey et al. (2023) - *Targets containerized environments to deploy cryptocurrency*	Treatment-
Cado Security	Threat Actors Hijack Cloud Services (Cado Security Labs, 2023).	

	-Cado reports a new tool 'legion," automating the exploitation of misconfigured web applications and hunting for .env files to retrieve credentials	

A summary of the threat actors and their behavioral description from the intelligence collection reveals that most of these threat actors' tactics focus on credential theft and privilege escalation. In the context of cloud security, this indicates a significant risk to cloud environments as threat actors target vulnerabilities in authentication mechanisms and exploit misconfigurations to gain unauthorized access and elevate their privileges within cloud infrastructures.

Threat actors can compromise cloud services, access sensitive data, and potentially disrupt operations by emphasizing credential theft and privilege escalation. This highlights the critical importance of robust security measures, continuous monitoring, and proactive threat detection strategies in cloud environments to mitigate these risks effectively. Table 4.2 below shows an effort to map the resulting TTPs from the threat intelligence reports to the actual tactic descriptions, tools, and API calls leveraged by the respective threat groups.

Table 4.2 Threat Actor TTP mappings to MITRE ATT&CK

Tactic	**Techniques & Tools / API Calls**	**Threat Actors**
Persistence	• *Techniques*: Additional Cloud Credentials, Device Registration • *Tools/API Calls*: AWS CLI commands (e.g., aws iam create-access-key, aws ec2 run-instances)	Scattered Spider, Cozy Bear, Cosmic Wolf

Privilege Escalation	• *Techniques*: Credential Access, Exploiting Misconfigurations • *Tools/API Calls*: Credential harvesting tools, AWS API calls (e.g., aws sts assume-role, aws iam attach-role-policy)	Team TNT, Cozy Bear, Cosmic Wolf
Credential Access	• *Techniques*: Unsecured Credentials, Credential Dumping • *Tools/API Calls*: Credential dumpers, AWS Secrets Manager (e.g., aws secretsmanager get-secret-value)	Cozy Bear, Cosmic Wolf
Discovery	• *Techniques*: Account Discovery, Resource Enumeration • *Tools/API Calls*: AWS CLI listing commands (e.g., aws ec2 describe-instances, aws iam list-users)	Team TNT, Cosmic Wolf
Lateral Movement	• *Techniques*: Use of Legitimate AWS APIs for Pivoting • *Tools/API Calls*: AWS SSM commands (e.g., aws ssm send-command)	Scattered Spider, Cosmic Wolf
Collection	• *Techniques*: Data Collection from Cloud Storage • *Tools/API Calls*: S3 commands (e.g., aws s3 ls, aws s3 cp)	Team TNT, Cosmic Wolf

4.2.2 Creating Testing Stories

In this project phase, testing stories or guides are created to facilitate the threat emulation stage of the research. It begins by extracting and mapping the threat actor TTPs to MITRE ATT&CK, as shown in Table 4.3. above. The mapping helps to refocus the emulation on specific techniques used by adversaries. MITRE also provides detection information for each technique that can aid in hypothesis generation. (National Cyber Security Centre, 2019)

The Threat Hunting Heatmap

A threat-hunting heat map serves as a valuable tool for assessing the effectiveness of the hunt team, as outlined by Cyb3rWard0g (2017). In this context, the heat maps illustrate the key techniques and areas of focus derived from our analysis of threat intelligence sources, specifically targeting Identity and Access Management (IAM) threats.

Figure 4.1 presents the threat-hunting heat map, which correlates threat actors from Cyber Threat Intelligence (CTI) with various tactics. Darker shades on the map indicate higher levels of activity. This visual representation supported the research in developing a comprehensive threat-hunting playbook, focusing on Persistence and Privilege Escalation techniques.

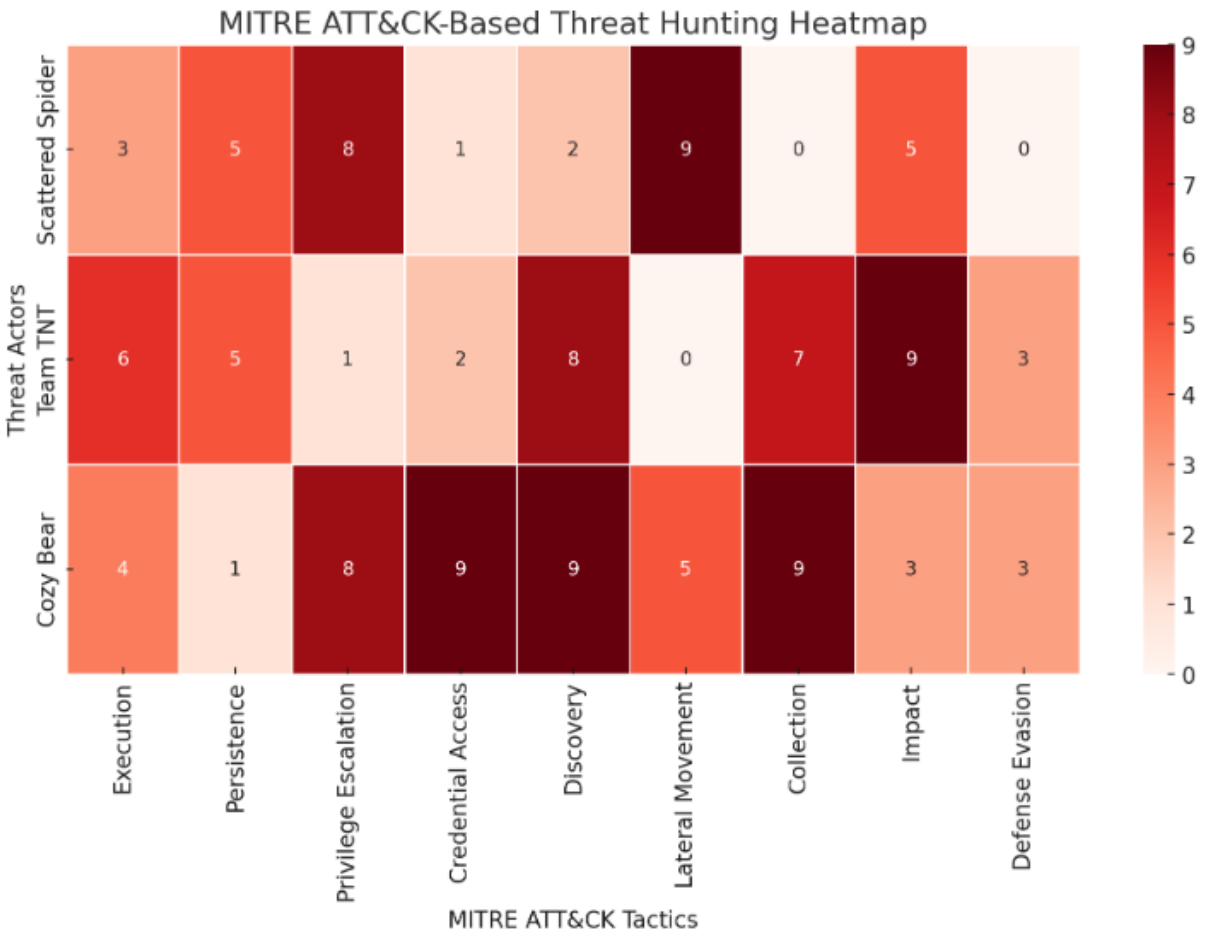


Figure 4.1: Threat hunting heatmap

Table 4.3 highlights the testing stories. The test was primarily focused on persistence and privilege escalation in cloud environments. Stratus tests, TDIR tests, and Cloud Goat offer different approaches to assessing these risks.

Table 4.3 Testing Stories

Technique	Stratus test	TDIR Test	Cloud Goat Test
Persistence and Privilege Escalation - Backdoor an IAM role - Create an administrative IAM user. - Create a login profile for an IAM user. - Create an IAM Roles Anywhere trust anchor. - Privilege escalation by attachment. - Privilege escalation by rollback. - Privilege escalation by rotation. - Unauthorized IAM credential usage	-Backdoor an IAM Role -Create an administrative IAM User. -Create a Login Profile for an IAM User. -Create an IAM Roles Anywhere trust anchor.	-Privs escalation by attachment. -Privs escalation by rollback. -Privs escalation by rotation.	Unauthorized IAM credential usage

4.2.3 Threat-Hunting Platform Design

The platform's design aimed to capture the threat-hunting platform's architecture, modules, components, and input data. The threat-hunting platform implementation replicates an AWS cloud environment with minimal network components to simplify environment setup. This consists of AWS CloudTrail for logging, Virtual Private Cloud (VPC), Elastic Compute Cloud (EC) instance, Simple Storage Service (S3) bucket for storage, and Identity and Access Management (IAM) users. The threat-hunting model's log flow from AWS CloudTrail is captured in Figure 4.2. The threat-hunting platform architecture components are illustrated below:

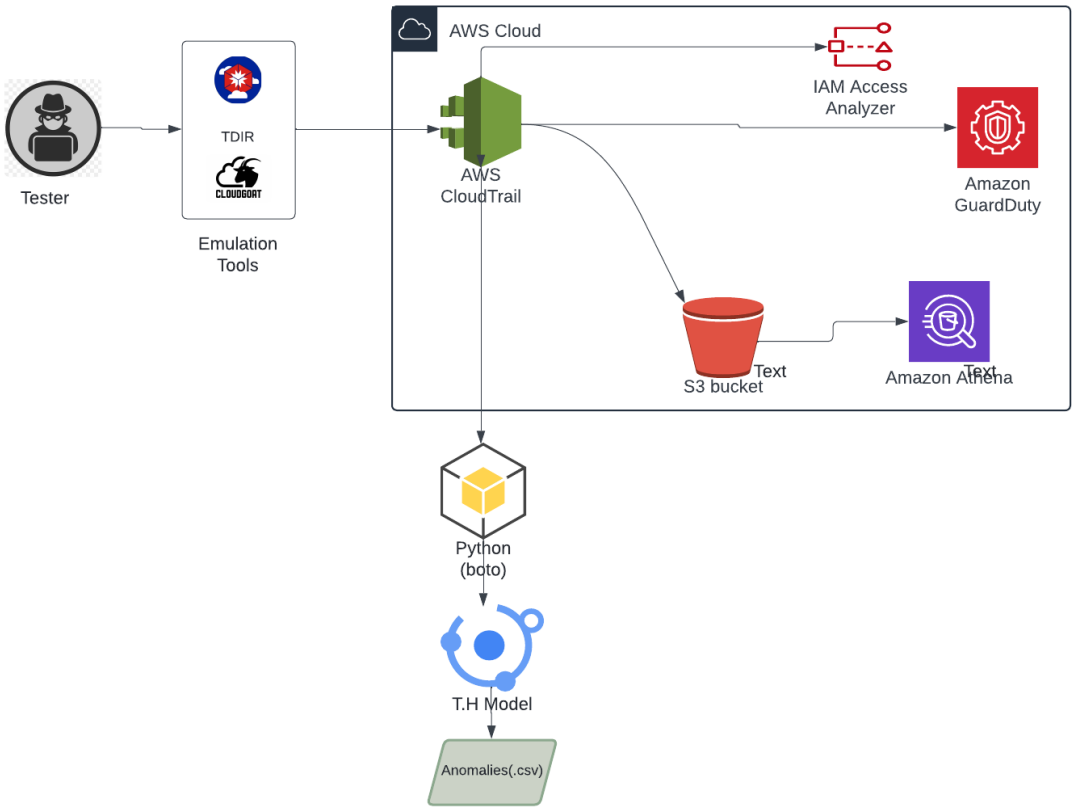


Figure 4.2: Threat Hunting Platform Design

a. Threat Emulation- Various tools are used for these attacks as captured in table 4.4, including Stratus Red Team, AWS Threat Detection and Response (TDIR), and CloudGoat. CloudTrail is the data log that captures management plane attack activities during the emulation exercise. The Stratus Red Team tool is open-source and simulates attacks across three primary cloud environments: AWS, GCP, and Azure (Tafani, 2022)

Table 4.4 Attack Emulation Tools

Tool	**Description**	**Purpose**
AWS TDIR	Set of best practices and guidelines for threat detection and incident response in AWS environments	Enhance security posture and detect and respond to security incidents in AWS.
Stratus Red Team	Red teaming platform for simulating real-world cyber-attacks and testing security defenses	Test and validate security controls, identify vulnerabilities and weaknesses in defenses
CloudGoat	Vulnerable by design AWS environment for educational and training purposes	Simulate security scenarios and practice identifying and mitigating security vulnerabilities.

4.2.4 Threat Emulation/Hunting Platform Flowchart

This research involves proactively searching for threats after the adversary emulation process. The threat-hunting platform approach comprises six stages described in Figure 4.3: scope, equipment, planning, weaponization, review, and execution.

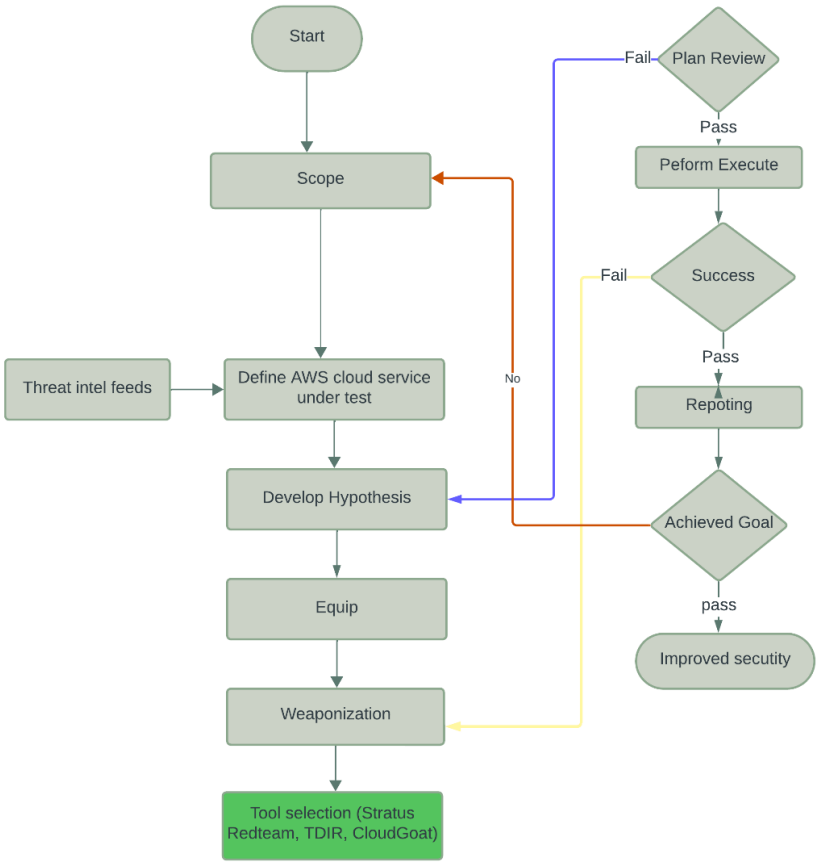


Figure 4.3:Threat Hunting Platform Flowchart

i. Scope phase: Every typical hunt considers the priority Intelligence Requirements that focus threat-hunting efforts on the organization's critical assets. This approach enables a more targeted and value-based threat-hunting expedition. In this case, the scoping was focused solely on the AWS IAM service, as this research focuses on identity-based misconfiguration that threat actors often exploit.

ii. Equip phase: This phase addresses questions regarding the previous build, including vulnerability information related to IAM services and associated attack mappings. It involves gathering data from various sources for analysis, such as identifying data sources and analytical TTPs.

iii. Weaponization phase: This phase uses exploits identified during the previous stage as part of the testing stories. Automated and manual tests are conducted here, focusing on persistence and privilege escalation attacks and targeting IAM services.

iv. The plan is reviewed by examining selected attack stories, such as the configuration setup on the emulation endpoint, before the actual execution. This also ensures that the model is already trained and launched to stream AWS logs concurrently with the attacks' execution. In this case, the boto3 client setup should be effectively configured to allow CloudTrail logs to be streamed.

v. The execution process was iterative. This was the final stage where the actual attacks occurred; each failed test was relaunched after reviewing the configuration parameters iteratively. Additionally, management plane events related to the attacks were streamed to the model, which classified identifiable threats and wrote the output to a CSV file for further analysis.

4.2.5 Threat Emulation Use Case Diagram

According to Gemino, A., and Parker, D. (2009), use case modeling is a popular system analysis and design approach. The use case diagram in Figure 7 illustrates how a security analyst interacts with the platform, beginning with the collection of threat intelligence, progressing to the creation of test scenarios based on this intelligence, and culminating in the execution of emulated attacks.

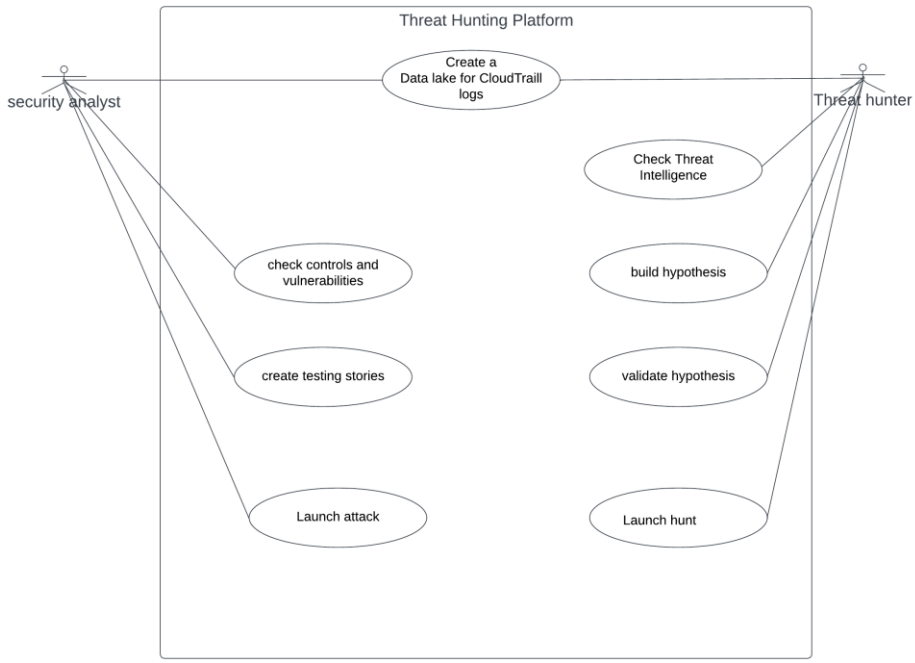


Figure 4.4: TH Use case Diagram

Figure 4.4 shows a generic use case design to illustrate platform core components, encapsulated as activities through the project's two phases, from the offensive to the counter-offensive. These include:

i. Uo1 Collecting threat intelligence - This threat intelligence is collected from cyber security solutions vendor-related reports. Precondition: Threat intelligence sources are accessible, i.e., vendor reports
ii. Uo2 Enable logging - AWS cloud trail logging is enabled, and logs are stored in an S3 bucket for threat analysis. **Precondition**: CloudTrail logging is enabled, and permission is granted to access logging features.
iii. Uo3 Creating testing stories - Creating attacks to test and exploit from insights from 1 and 2. Launching the attack—carrying out the attack using the Stratus Red team, CloudGoat, and TDIR. **Precondition**: The attack scenarios have been created based on insights from threat intelligence and vulnerability assessments.
iv. Uo4 Building hypothesis - Create a hypothesis of a possible attack**. Precondition**: Relevant data from previous attacks and threat intelligence is available for analysis to inform hypothesis development.
v. Uo5 Validating hypothesis- verifying the existence of relatable TTPs based on the launched attacks. Precondition: The attack has been launched, and corresponding logs from AWS CloudTrail are available for analysis to verify TTPs.

vi. Uo6 Launch hunt- start the hunting process guided by detections from the threat model. **Precondition**: Detection rules based on the threat model have been established, and relevant data sources and logs are available for analysis during the hunt.

While a security analyst can perform all these activities, a threat hunter is introduced explicitly in the use case design as a participant in the threat-hunting phase to delineate the proactive and counter-offensive aspects of the research process.

4.3 Counter-Offensive Phase: Detection, Modeling & Hunting

This phase focuses on the research's defensive aspect, detailing how the obtained public threat log dataset was trained to develop the threat hunting model for anomaly prediction.

4.3.1 Threat-Hunting Model Sequence Diagram

Swain et al. (2010) describe sequence diagrams as essential for modeling a system's behavioral aspects. They show a set of interacting objects and the messages exchanged among them. Figure 4.5 below shows the TH sequence diagram.

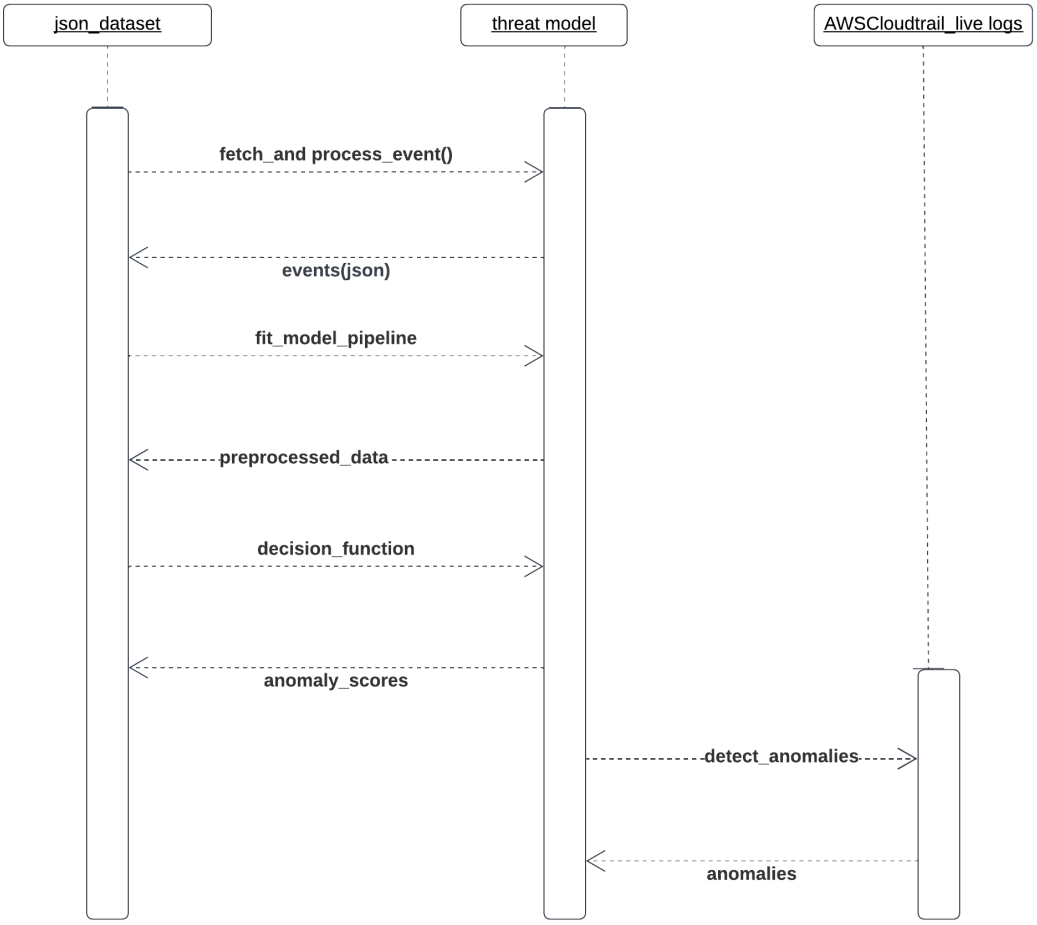


Figure 4.5: Threat Model Sequence Diagram

The model is based on the Isolation Forest Algorithm for predicting anomalies. The data preparation uses a label encoding algorithm that converts categorical features, such as event names, resource types, and user identities, into a numerical format during preprocessing. This data is basically transformed in a way that can be used in the model. The overarching concept of this threat-hunting (TH) model operates akin to a funnel, whereby it predicts outlier events derived from a collection of streamed AWS CloudTrail events. These outliers are recognized

as anomalies and are subsequently recorded in a CSV file for further analysis, thereby facilitating the validation of specific hypotheses. Thus, the fundamental notion of the TH model is to serve as a funnel, anticipating outlier events from a compilation of streamed CloudTrail events, which can then be subjected to additional analysis to corroborate specific hypotheses.

4.3.2 Threat Hunting Model Components

i. **Data Source-** AWS cloud trails JSON logs provided by Scot Piper for AWS research
ii. **Preprocessing-** The preprocessing is achieved by extracting relevant fields: user agent, event type, event name, AWS region, source IP, and user identity type. These fields are used as part of the label-encoded categorical features for building the threat model.
iii. **Model Pipeline-** The model used in this case is Isolation Forest, an anomaly detection algorithm effective for identifying unusual patterns in data. A Model pipeline is set up after the preprocessing phase, where feature extraction is done based on the Isolation Forest algorithm
iv. **Anomaly scoring**- This is achieved based on Isolation Forest predictions, where categorization scores are (high, medium, low, and standard).
v. **Fetching Live Logs-** Once the model is trained on the dataset, it fetches live logs for classification/prediction from AWS CloudTrail using lookup_events
vi. **Anomaly detection**- the model uses a trained pipeline to generate event anomaly scores.
vii. **Output**- A .csv file saves counts of anomalies with their anomaly score.

The threat-hunting model focuses primarily on Identity and Access Management (IAM). IAM threats involve identifying and mitigating user roles, groups, and credentials risks, as attackers often exploit IAM vulnerabilities to gain persistent access to networks and escalate their privileges. Common tactics include creating malicious IAM users to maintain access, establishing backdoor IAM users, and utilizing enumeration techniques to discover IAM roles. Both IAM vulnerabilities pose substantial security threats, emphasizing the critical need for robust monitoring and management practices within AWS environments to prevent exploitation by threat actors. The model is designed to retrieve, preprocess, analyze, and detect anomalies in AWS CloudTrail events, focusing on IAM activities and providing insights into potentially suspicious activities.

4.4 Building Hypothesis

Every hunting session should start with a hypothesis about the attacker's behavior, aiming to validate this idea in the real world using available data sets. This hypothesis must be testable and grounded in situational awareness derived from threat intelligence, allowing for either its confirmation or rejection (National Cyber Security Centre, 2019). This research concentrated on a specific hypothesis related to threat detection, employing a technique-based approach grounded in the MITRE framework. A series of hunt sessions were conducted based on these three hypotheses to validate the behaviors of current threat actors.

i. H1: T1078.004 - Valid accounts in cloud systems can enable adversaries to execute actions for initial access, persistence, privilege escalation, or evasion of defenses. Unauthorized actors may exploit exposed AWS IAM access keys to gain access to AWS resources, potentially leading to data exfiltration, resource manipulation, or service disruption.

ii. H2: T1098.003—Adversaries may manipulate accounts to gain or enhance their access to victim systems. To do this, they must have sufficient permissions on those systems or within the domain. Once they have gained access, adversaries can assign additional roles or Permissions to a compromised cloud account ensure attackers maintain persistent access and can escalate their privileges within the environment. For example, an attacker might assign an Administrator role to an IAM user, solidifying their foothold in the compromised environment and enabling further exploitation.

iii. H3: T1136.003 - Adversaries can create a cloud account to secure ongoing access to victim systems. Once the account is set up, they manipulate it to ensure persistent access and the ability to tap into additional resources. This manipulation might involve adding more cloud credentials or assigning extra cloud roles, enhancing their capability to control and exploit the victim's environment effectively.

4.5 Conclusion

Chapter 4 described developing and validating a threat-hunting platform focusing on Identity and Access Management (IAM) threats. It detailed the architecture using AWS services like CloudTrail, GuardDuty, and IAM Analyzer for logging and threat detection, alongside tools like Stratus Red Team and CloudGoat for simulating attacks. The workflow consisted of phases such as scoping, equipping, weaponization, review, and execution, ensuring comprehensive

Chapter 5: Implementation and Findings

5.1 Introduction

This chapter demonstrates implementing and testing the threat model's threat-hunting capability. Testing the threat model was crucial to ensuring the effectiveness of the research's offensive and counter-offensive aspects. This chapter builds on the previous chapters by providing specific details about the environment setup, platform testing, and results.

5.2 Overview of the Threat Model

The threat-hunting (TH) model employed a funnel-like methodology, processing CloudTrail events to predict anomalous events. Anomalies were identified by comparing them to established behavior patterns, and these anomalies were systematically recorded as output in a .csv file. This process facilitated the validation of specific hypotheses and encompassed the following key functions and modules components:

1. **Log Management**
    i. load_json_logs(file_pattern): Aggregates and loads the training dataset from multiple JSON files.
    ii. preprocess_logs(records): Transforms raw logs into a structured format.
2. **Data Processing**
    i. model (random_state=42, contamination=0.1, n_estimators=250): Constructs an anomaly detection pipeline by defining the training parameters
    ii. fetch_and_process_cloudtrail_events (cloudtrail_client, save_path='processed_cloudtrail_logs. json'): Fetches AWS CloudTrail events for processing while also saving them as a JSON file
3. **Model Management**
    i. save_model- Saves a machine learning model to disk.
    ii. load_model- Loads a previously saved model from disk.

These functions create a comprehensive framework for managing the T.H. machine learning model. The load_json_logs function in Figure 5.1 aggregates a JSON-formatted log data source, allowing for efficient data consolidation and analysis.

```python
# 2. Data Loading
def load_json_logs(file_pattern):
    """Load JSON logs from files matching a pattern."""
    records = []
    for file_path in glob.glob(file_pattern):
        try:
            with open(file_path, 'r', encoding='utf-8') as file:
                data = json.load(file)
                records.extend(data.get('Records', []))
        except Exception as e:
            logging.error(f"Failed to load or parse JSON file {file_path}: {e}")
    return records
```

Figure 5.1: Loading input data function

The preprocess_logs function transforms raw log entries into a structured format (data frame) suitable for further analysis or machine learning applications. Figure 5.2 Shows the output of the preprocessed data frame.

	accountId	eventName	eventTime	awsRegion	eventType	eventSource	sourceIPAddress	userIdentityArn	e
1858033	811596193553	DescribeDBSubnetGroups	2020-07-06T07:30:41Z	eu-west-1	AwsApiCall	rds.amazonaws.com	152.250.116.241	arn:aws:iam::811596193553:user/Level6	
543591	811596193553	RunInstances	2019-08-21T12:57:49Z	ap-south-1	AwsApiCall	ec2.amazonaws.com	5.205.62.253	arn:aws:iam::811596193553:user/Level6	
27810	811596193553	GetPolicyVersion	2017-05-26T22:59:51Z	us-east-1	AwsApiCall	iam.amazonaws.com	255.253.125.115	arn:aws:sts::811596193553:assumed-role/Securit...	
746938	811596193553	RunInstances	2019-08-21T20:45:12Z	ca-central-1	AwsApiCall	ec2.amazonaws.com	5.205.62.253	arn:aws:iam::811596193553:user/backup	
1376537	811596193553	RunInstances	2019-08-22T22:56:52Z	eu-central-1	AwsApiCall	ec2.amazonaws.com	5.205.62.253	arn:aws:iam::811596193553:user/backup	

Figure 5.2: Result of preprocessed record into a data frame

During data processing, as seen in Figure 5.2, categorical features are defined and encoded using a label encoder algorithm into the data frame as part of feature engineering.

Label encoder

```python
from sklearn.preprocessing import LabelEncoder
categorical_cols = ['accountId','eventName', 'awsRegion', 'eventType','eventSource', 'sourceIPAddress', 'userIdentityArn', 'errorMessage', 'errorCode', '
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    features_df[col] = le.fit_transform(features_df[col].astype(str))
    label_encoders[col] = le
```

Figure 5.3: Features Encoding

Figure 5.3 shows how a threat-hunting model is fitted to the training data and saved to a file. The model is initialized with the parameters random_state, contamination, and n_estimators.

Train and Save model

```
# fitting the model
model = IsolationForest(random_state=42, contamination=0.1, n_estimators=250)
model.fit(X_train)

# Save the model to a file
with open('isolation_forest_model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

Figure 5.4: Fitting and Saving the Model

i. **random_state=42:** ensures reproducibility by setting a seed for random number generation
ii. **contamination=0.1:** specifies the proportion of outliers in the dataset
iii. **n_estimators=250**: sets the number of base estimators (trees) in the ensemble to 250.

The parameter `n_estimators` refers to the number of Isolation Trees utilized within the ensemble method. The Isolation Forest research paper (Datacamp, 2024) suggests using a value of 100, which yielded effective results across various datasets. Additionally, `contamination` indicates the expected percentage of data points considered anomalies. The Isolation Forest model is trained using the dataset to learn the underlying patterns and identify outliers. Once trained, the model is stored as `isolation_forest_model.pkl` and utilized later for making predictions on new, live CloudTrail logs.

The function `fetch_and_process_cloudtrail_events ()` processes AWS CloudTrail logs by converting them into a Data Frame format while managing potential errors during data retrieval. This function performs exception checks for internal AWS API calls based on the event source and IP addresses. Finally, the function returns the `processed_data` Data Frame, which includes all the processed CloudTrail events.

```python
# Define your internal AWS services here
INTERNAL_AWS_SERVICES = {'access-analyzer.amazonaws.com','config.amazonaws.com', 'guardduty.amazonaws.com','athena.amazonaws.com','securityhub.amazonaws.

def fetch_and_process_cloudtrail_events(cloudtrail_client, save_path='processed_cloudtrail_logs.json'):
    processed_events = []
    next_token = None

    try:
        while True:
            response = cloudtrail_client.lookup_events(NextToken=next_token) if next_token else cloudtrail_client.lookup_events()
            events = response.get('Events', [])
            next_token = response.get('NextToken')

            for event in events:
                try:
                    eventData = json.loads(event.get('CloudTrailEvent', '{}'))
                    if (eventData.get('eventSource') in INTERNAL_AWS_SERVICES or
                        eventData.get('sourceIPAddress') in INTERNAL_AWS_SERVICES or
                        eventData.get('userAgent') in INTERNAL_AWS_SERVICES or
                        eventData.get('eventName') == "LookupEvents"):
                        continue  # Skip this event

                    processed_event = {
                        'accountId': eventData.get('recipientAccountId', 'Unknown'),
                        'userAgent': eventData.get('userAgent', 'Unknown'),
                        'eventType': eventData.get('eventType', 'Unknown'),
                        'eventSource': record.get('eventSource', 'Unknown'),
                        'eventName': eventData.get('eventName', 'Unknown'),
                        'awsRegion': eventData.get('awsRegion', 'Unknown'),
                        'sourceIPAddress': eventData.get('sourceIPAddress', 'Unknown'),
                        'userIdentityArn': eventData.get('userIdentity', {}).get('arn', 'Unknown'),
                        'errorMessage': eventData.get('errorMessage', 'None'),
                        'errorCode': eventData.get('errorCode', 'None'),
                    }
                    processed_events.append(processed_event)
                except json.JSONDecodeError as json_err:
                    logging.error(f"JSON decode error while processing CloudTrail event: {json_err}")
                except Exception as e:
                    logging.error(f"Error processing CloudTrail event: {e}")

            if not next_token:
                break

    except Exception as e:
        logging.error(f"Failed to lookup CloudTrail events: {e}")
        return pd.DataFrame()  # Return an empty DataFrame on failure
```

Figure 5.5: Processing AWS CloudTrail events

Figure 5.6 shows a plotted visualization of the anomaly scores' distribution based on the training data's decision function. A decision function is a numerical score indicating the degree of anomaly; lower values indicate more extreme anomalies.

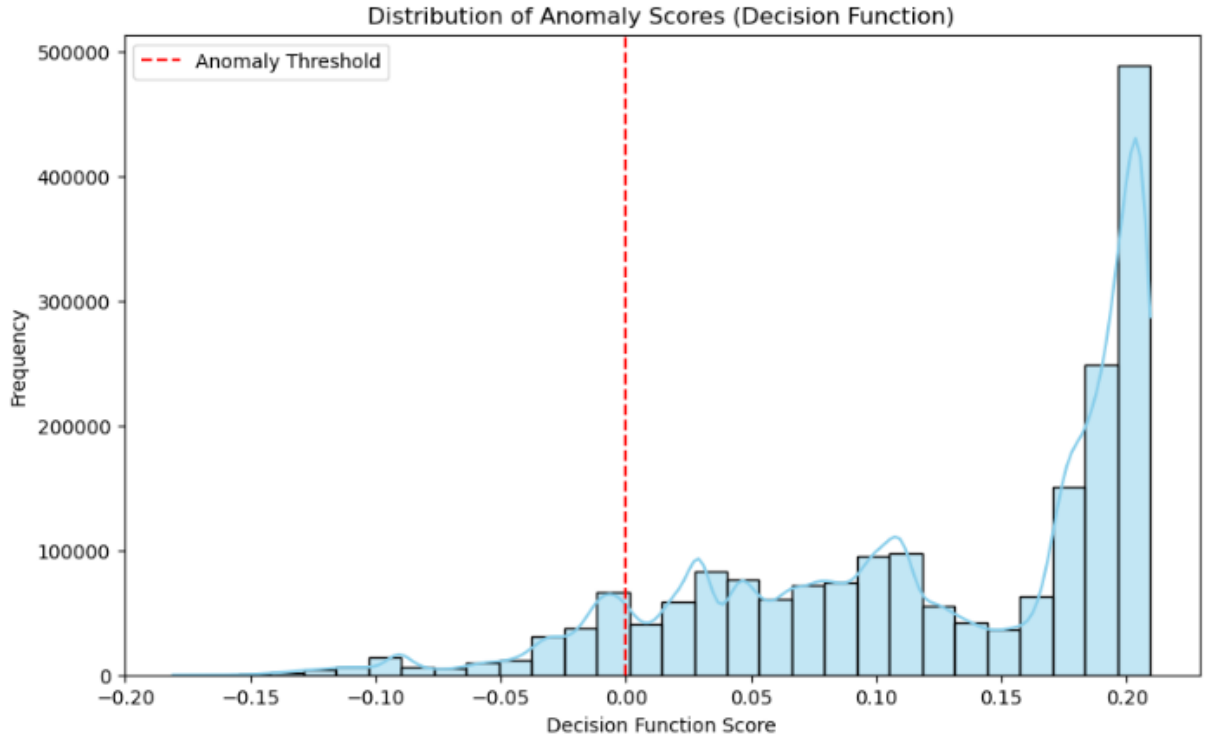


Figure 5.6:Distribution of the decision function scores for the train data

5.3 System Implementation

The experiment setup phases were implemented in a multifaceted environment to distinguish the nature of the test between offensive and counter-offensive actions. The project implementation follows this structure:

i. The Emulation phase consists of attacks launched against an AWS environment.
ii. Logs are captured in AWS Cloud Trail and stored in an S3 bucket for later retrieval, and the Boto3 client described in Chapter 4 is utilized to stream events from AWS Cloud Trail to the TH model.
iii. The TH model predicts anomalous events and outputs them to a .csv file.

For the emulation test, the emulation endpoint is a Debian operating system running Kali Linux. The AWS cloud environment was the test case, with logs collected in an S3 bucket.

5.3.1 Hardware Environment

The research used a virtual Box machine as the emulation endpoint with the following specifications; below is the minimum specification for meeting the research requirements.

i. Memory 2 GB
ii. Processor 4 cores
iii. Hardware 80 GB
iv. Virtualization Hardware

5.3.2Software Environment

i. AWS CLI
ii. Python
iii. Terraform for IAC deployment of att&ck scenarios
iv. Amazon Athena
v. AWS SDK bot

5.4 Executions of the Offensive Phase

The tests were conducted by executing tests against our lab environment, the tests covered. The tests were divided into automated and manual tests covering tactics: **Persistence, Privilege Escalation, and Collections.**

5.4.1 Automated Test (Stratus Tests)

The execution of the Stratus Red Team test commenced with the downloading of the Stratus Red Team binary from GitHub. Subsequently, the AWS profile and region are exported as environment variables on the emulation endpoint, initiating the execution process. The following phases execute the Stratus Red Team test:

1. Warmup: A terraform script spins up the prerequisite infrastructure without detonating. All attacks are, by default, in a cold state
2. Detonate: Executes the attack technique when the infrastructure is warm.
3. Revert: Reverts to a warm state where an attack can be detonated again.
4. Cleanup: Removes all prerequisite infrastructure from the environment, returning the infrastructure to a cold state.

Four tests are conducted in this phase, as shown in Table 5.1.

Table 5.1: Stratus Tests executed

Test No.	**Description**
Test 1	Simulates a login to the AWS Console for an IAM user without multi-factor authentication (MFA) **Command:** /stratus detonate aws.initial-access.console-login-without-MFA
Test 2	Establishes persistence by creating an IAM Roles Anywhere trust anchor. The IAM Roles Anywhere service allows workloads that do not run in AWS to assume roles by presenting a client-side X.509 certificate signed by a trusted certificate authority called a "trust anchor. **Command:** /stratus detonate aws.persistence.roles anywhere-create-trust-anchor
Test 3	Establish persistence by creating a login profile for an existing IAM user. This allows an attacker to access an IAM user intended for programmatic use through the usual AWS console login process. **Command:** ./stratus detonate aws.persistence.iam-create-user-login-profile
Test 4	Establishes persistence by creating a new IAM user with administrative permissions. **Command:** /stratus detonate aws.persistence.iam-create-admin-user
Test 5	Establishes persistence by backdooring an existing IAM role, assuming it is from an external AWS account. **Command:**/stratus detonate aws.persistence.iam-backdoor-role
Test 6	Establishes persistence by creating an access key on an existing IAM user. **Command**: ./stratus detonate aws.persistence.iam-backdoor-user

Figure 5.7 shows tests already detonated against an AWS environment. An executed test automatically changes the status to detonate.

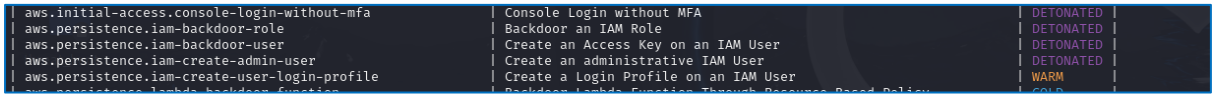


Figure 5.7: Checking Test Status

5.4.2 AWS TDIR Test

These are tests provided for AWS incident response preparedness. The scenario of focus in this phase is emulating a security event in which an unauthorized user discovers and utilizes IAM credentials. The Environment is set up using a CloudFormation template as shown in Figure 5.8.

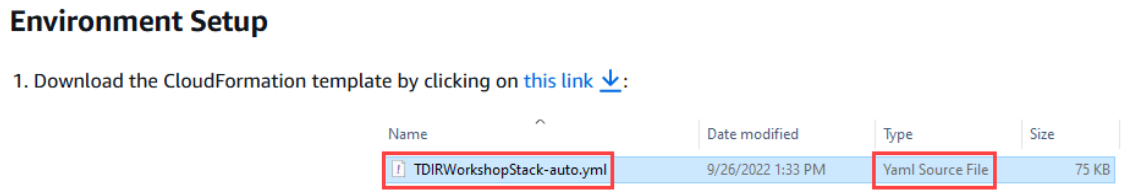


Figure 5.8: Downloading the CloudFormation Template

The CloudFormation template is then used to create a stack on the AWS console of my test environment, as shown in Figure 5.9.

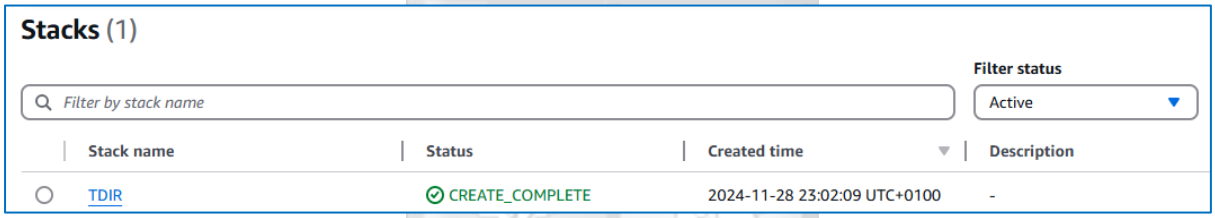


Figure 5.9: Complete environment setup for AWS TDIR test

Upon completion of the setup, emulation is achieved by invoking a Bash script from either the AWS Cloud Shell or the emulation endpoint. The script then performs automated attack activities, and after the script, an exposed access key ID is generated, which is utilized for detecting activities. Both the Emulation script and CloudFormation are downloaded from the AWS workshops site.

The sequence of actions taken during the emulation processes is captured in Table 5.2.

Table 5.2: The IAM exploitation attack chain

steps	Action	Description
1	Initiate Attack Simulation	Start the simulation to emulate an attack scenario.
2	Retrieve AWS Account ID	Obtain the AWS Account ID for further actions.
3	Create IAM Policy	Develop an IAM policy that facilitates unauthorized access.
4	Simulate Multiple IAM Users	Create several IAM users to demonstrate potential credential exposure.
5	Attach Read-Only Policies	Attach read-only policies to simulate limited access scenarios.
6	Generate Access Keys	Produce access keys for the created IAM users, enabling programmatic access.
7	Run EC2 Instances	Launch EC2 instances as part of the attack simulation to show resource exploitation.
8	Create S3 Buckets	Establish S3 buckets to illustrate data storage and potential data exfiltration points.
9	Expose Access Key ID	Expose Access Key IDs as a demonstration of credential leakage.

5.4.3 Manual Test (Cloudgoat)

CloudGoat is a tool developed by Rhino Security Labs that enables the creation and completion of various scenarios for exploring AWS environments, identifying vulnerabilities, and exploiting them to achieve the desired outcome. In this attack scenario, exploiting a vulnerability in the AWS IAM through IAM privilege escalation by rollback allows a low-privileged user to gain full administrative access. The attacker gains access to a previous IAM policy version, which allows for full administrative privileges, and subsequently uses SetPolicyVersion to update the current policy to a policy with elevated privileges, resulting in

a privilege escalation exploit. Figure 5.10 shows the output of the fully deployed scenario, with the user details.

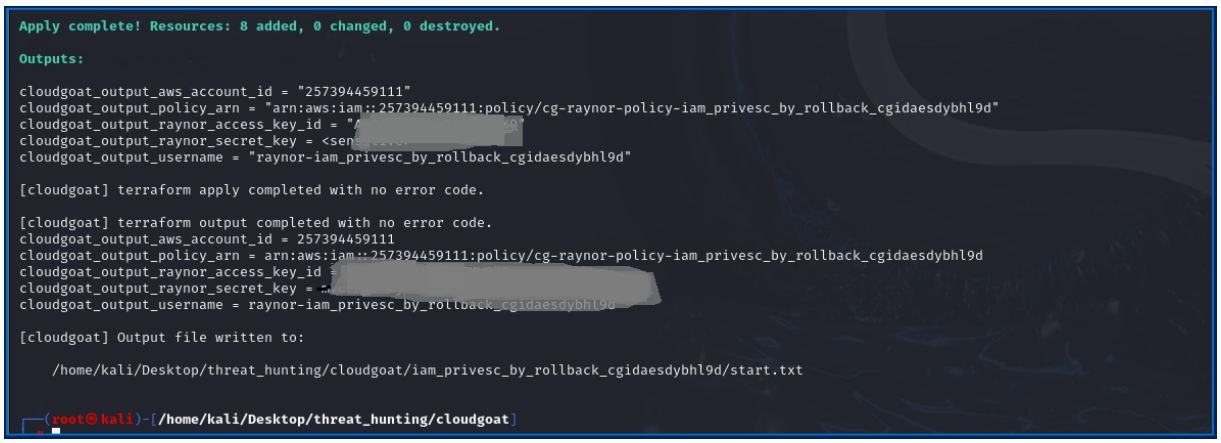


Figure 5.100: Cloud IAM privsec by rollback complete deployment.

The manual emulation begins by setting the user profile and following the attack path in Figure 5:11 to exploit the system successfully.

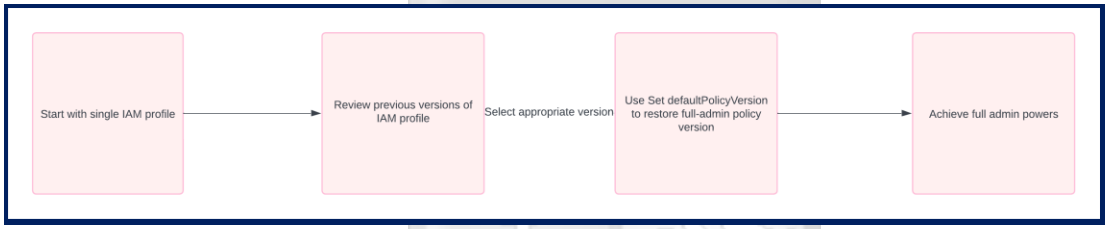


Figure 5.11: Scenario 1 IAM privilege escalation attack path.

5.5 Counter-Offensive Phase

A central log collection to facilitate log analysis was a prerequisite for this second research stage. The threat-hunting process starts in this phase, from hypothesis building to validation. The successful accomplishment of these activities was considered a successful hunt.

5.5.1 Log Collection

To ensure visibility into the threat emulation activities, we established a CloudTrail in AWS with S3 management events enabled. This "thesis trail" was configured to store its logs in the "thesis test" S3 bucket, as illustrated in Figure 5.12.

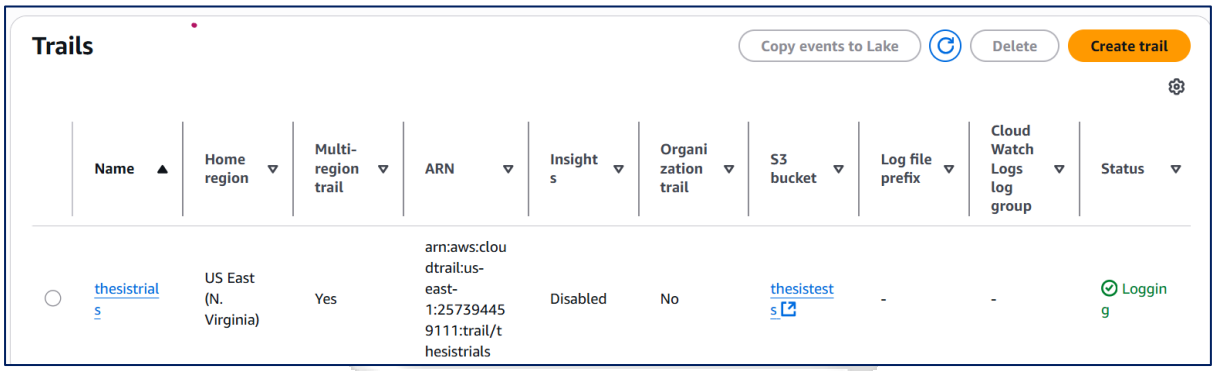


Figure 5.12: Visibility through Log collection

To hunt and validate my hunts against the threat model, Amazon Athena, and Amazon Glue were used to create a database for querying and testing analytics for ad hoc and structured hunts.

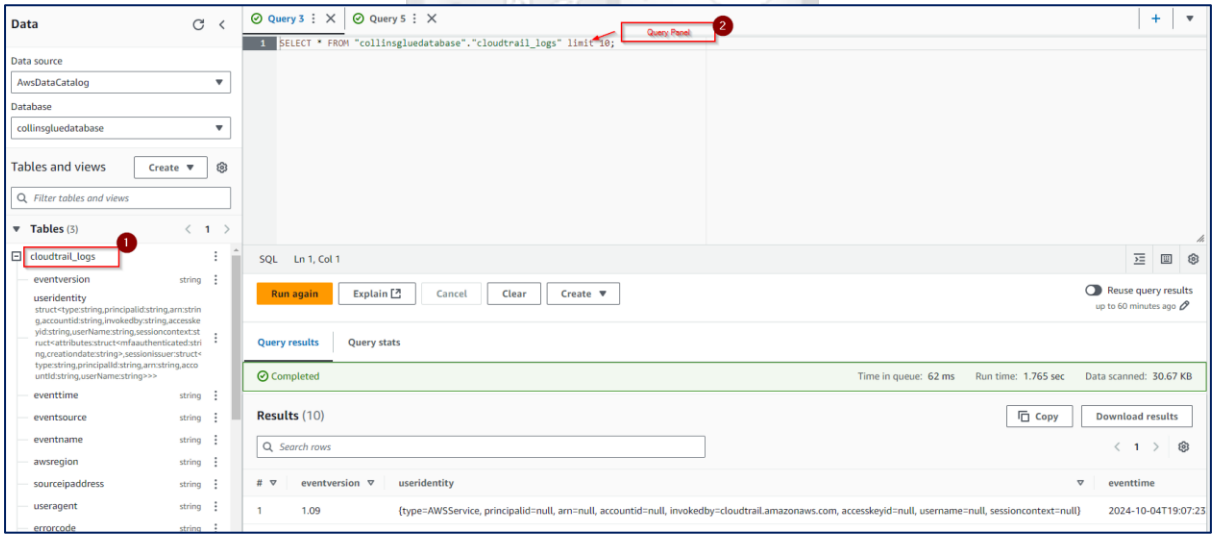


Figure 5.13: Amazon Athena Queries

5.6 Result analysis the Hypothesis Driven TH

Every threat-hunting session begins with a hypothesis about an attacker's behaviour, which we then seek to verify using real-world data sources. The objective is to confirm or reject this hypothesis, much like the scientific research process. If the validation statement is false, it indicates that the query did not yield any relevant results, leaving the specific validation question unanswered. Conversely, if the statement is true, it suggests that the query produced results, thereby confirming the threat-hunting question. Successful validation means the model accurately identified anomalous events related to the hypothesis. This analysis is conducted on the output “detected_cloudtrail_anomalies.csv." This .csv file results from classifying anomalous events by the TH model. Figure 5.4. shows the analysis of the categorized events generated from the threat offensive phase of this research.

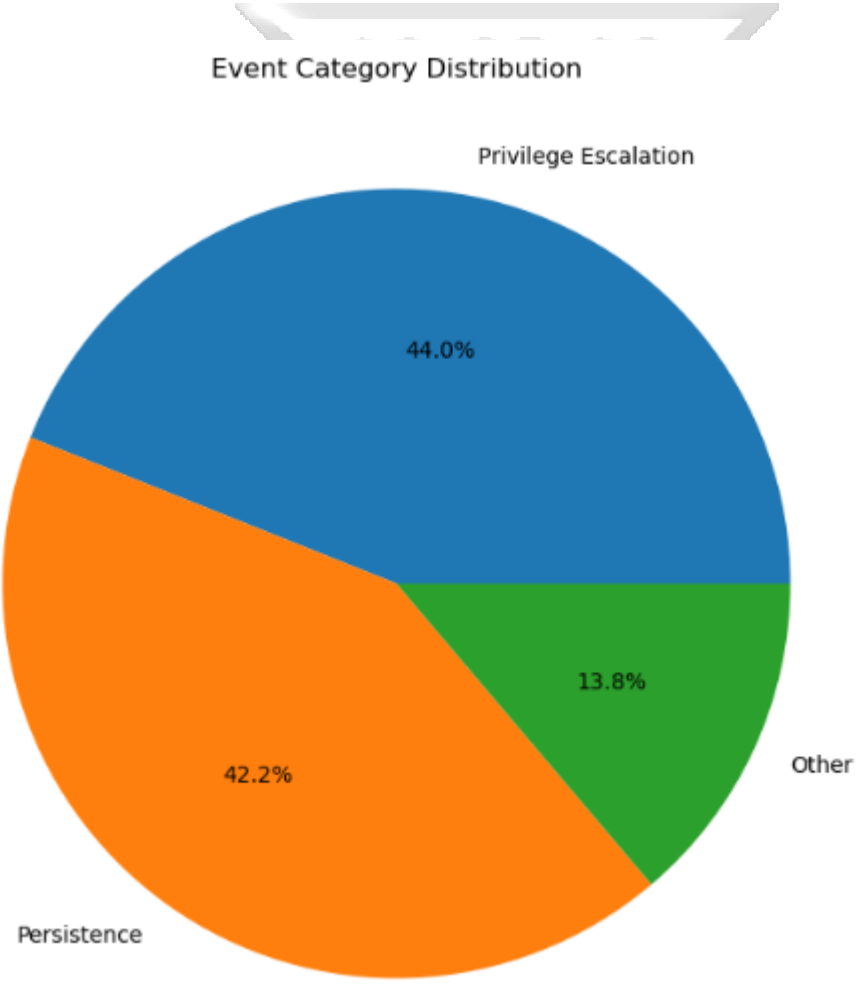


Figure 5.14: Event category distribution by Tactic

5.6.1 Hypothesis 1

This hypothesis focuses on T1078.004: Adversaries may exploit compromised cloud credentials to gain unauthorized access to AWS services. This technique enables attackers to maneuver within the cloud environment, potentially resulting in persistence and privilege escalation (MITRE, 2024). Figure 5.15 shows a description of T1078.004.

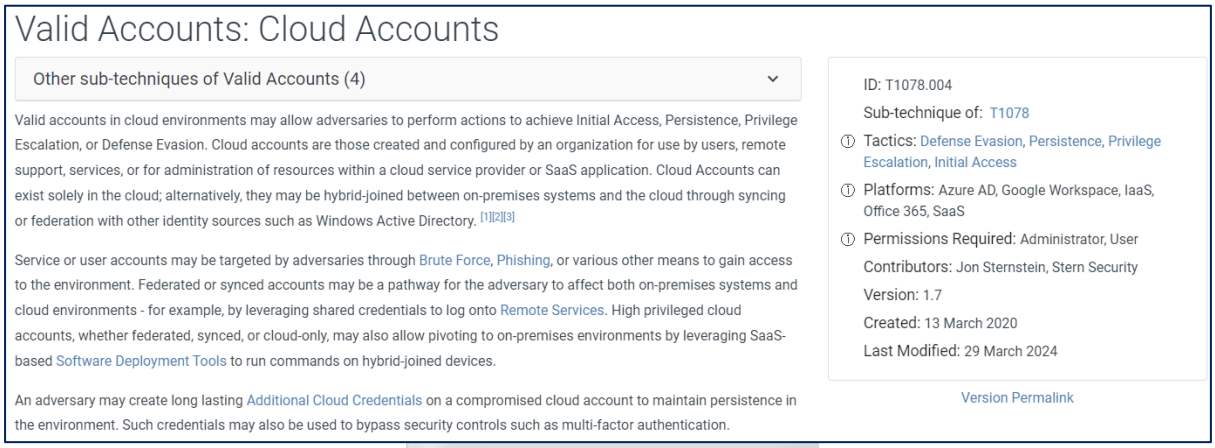


Figure 5.14: Hypothesis created based on ID: T1078.004

This research relied on the predicted results of the threat-hunting model to validate this hypothesis. The hypothesis was validated based on the questions outlined below. For a structured approach, a naming convention for the validation criteria is used, denoted as "*hunt. validation*" (h.v).

- [**h1.v1**] Check for High volumes of access-denied errors from a specific identity.
- [**h1.v2**] Checking for console login without MFA
- [**h1.v3**] Checking for access-denied errors from creating an IAM user across multiple accounts.

Analytic queries were developed as a threat-hunting technique to test our hypothesis. This involved further analysis of the model output "*detected_cloudtrail_anomalies.csv*." Ensuring all log artifacts are captured to help with correct attribution. Table 5.3 shows the validation analysis query of the hypothesis [**h1.v1**]

Table 5.3: Validation analysis query of h1.v1

```
import pandas as pd
import os
```

```
# Specify the path to your CSV file
file path = os.path.join(os.getcwd(), 'detected_cloudtrail_anomalies.csv')

# Read the CSV file into a DataFrame
try:
    df = pd.read_csv(file_path)
Except Exception as e:
    print(f"Error reading the file: {e}")
    exit()

# Filter the DataFrame for IAM and S3 events with a non-null errorCode
filtered_df = df[
    df['eventSource'].isin(['iam.amazonaws.com']) &
    df['errorCode'].notnull() & (df['errorCode'] != '')
]

# Group by userIdentityArn and count the number of unique error codes per user
error_counts = filtered_df.groupby('userIdentityArn')['errorCode'].nunique().reset_index(name='error_count')

# Sort by error count in descending order and print the top 10 users
top_10_users = error_counts.sort_values(by='error_count', ascending=False).head(10)
print("Top 10 users with the most distinct error events:")
print(top_10_users)
```

[h1.v1] This analysis identifies instances where IAM users generate many API calls accompanied by error messages, which may indicate potential unauthorized activities. The accompanying graph visually represents user identities associated with a high frequency of access-denied incidents. From the graph, the users "*raynor_iam_privsec*", “tdir-workshop-wolf-dev", and “Kerrigan” show particularly notable events, which may warrant further investigation to uncover the possible cause of increased errors in their API calls. The primary

objective of this analysis within the scope of threat hunting is to expeditiously recognize potential anomalous behaviors that can act as a pivot point for investigation by concentrating on user activities that yield error codes, thereby serving as an initial point of inquiry in the investigation. Figure 5.16 shows identities with the highest number of error messages.

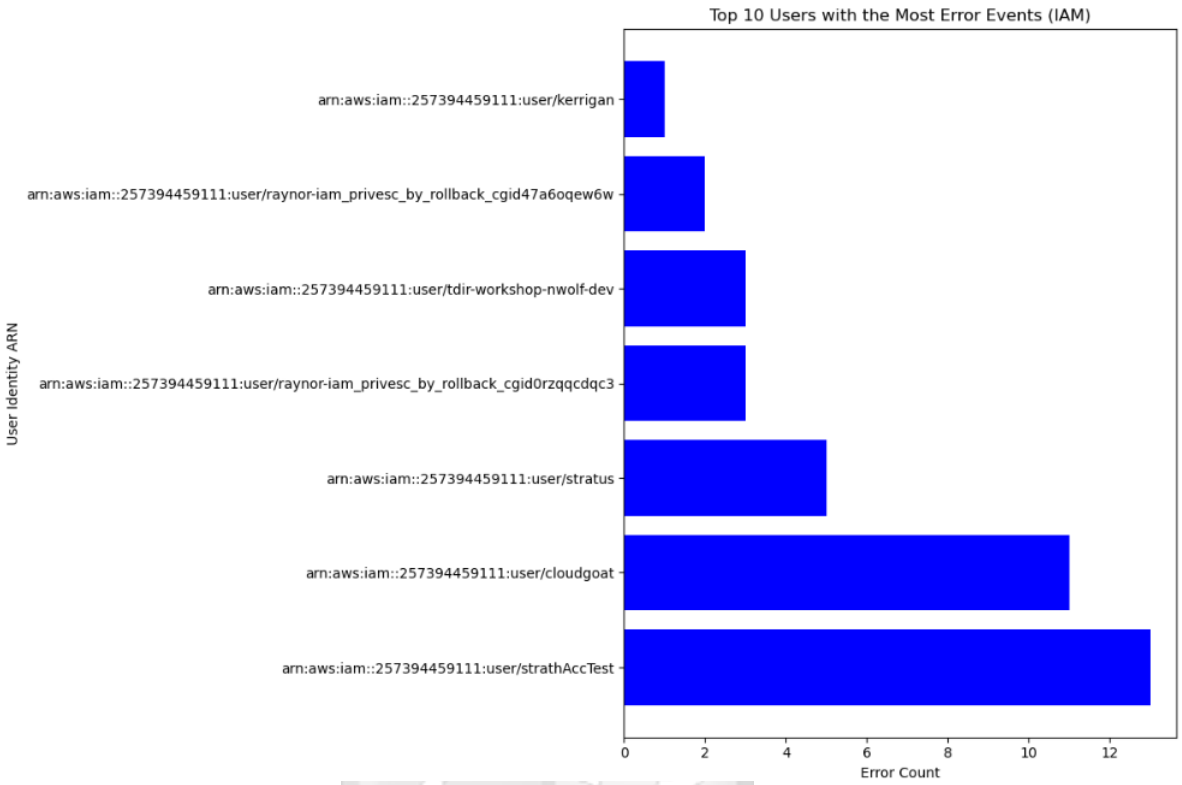


Figure 5.156: UserIdentities with a high number of error messages.

Narrowing down the scope to “AccessDenied” errors, as in Figure 5.17, provides valuable context for the initial investigation. Understanding the event resulting from these errors gives the researcher insights into the nature of the activities. This information helps us correlate different events and identify those that contextually may need deeper investigation.

	userIdentityArn	failed_logins	unique_events	userName	sourceIPAddress	eventName
1	arn:aws:iam::257394459111:user/tdir-workshop-nwolf-dev	3	3	Unknown	54.204.116.47	ListPolicies
2	arn:aws:iam::257394459111:user/tdir-workshop-nwolf-dev	3	3	Unknown	54.204.116.47	ListUsers
3	arn:aws:iam::257394459111:user/tdir-workshop-nwolf-dev	3	3	Unknown	54.204.116.47	ListRoles
4	arn:aws:iam::257394459111:user/raynor-iam_privesc_by_rollback_cgid47a6oqew6w	2	2	Unknown	149.102.231.84	ListPolicyVersions
5	arn:aws:iam::257394459111:user/raynor-iam_privesc_by_rollback_cgid47a6oqew6w	2	2	Unknown	149.102.231.84	ListUsers

Figure 5.16: Enhanced analysis of additional fields to better correlate failed activities

The second part of the analysis aimed to validate [**h1.v2**] by checking for console login without MFA as part of initial access. However, the model could not explicitly capture MFA usage due to the limited extracted features. While the analysis identified suspicious console logins, it

could not definitively confirm the absence of MFA. These console logins are captured in Figure 5.18. The discrepancy observed highlights the importance of considering additional factors and refining the model to detect such events accurately in the future.

	accountId	eventName	eventTime	awsRegion	eventType	sourceIPAddress	userIdentityArn
1	257394459111	ConsoleLogin	2024-12-02T17:55:16Z	us-east-1	AwsConsoleSignIn	109.81.160.83	l:user/stratus-red-team-nmfalu-yailehkato
2	257394459111	ConsoleLogin	2024-11-27T07:28:11Z	us-east-1	AwsConsoleSignIn	45.145.155.2	user/stratus-red-team-nmfalu-nokeekqvzj

Figure 5.18: Initial access Console Login activities

Only two validation questions were successfully answered in Hypothesis 1, as seen in *Table 5.4:H1*. Validation questions were curated to guide the process of testing hypothesis developed as part of the research. A successful validation of the hypothesis formed the basis for furthering the hunt. When the validation statement is false, it means the query yielded no results, whereas when true, the query showed results that validating the hypothesis.

Table 5.4: H1 Validation results

	Hypothesis Validation	
[h1.v1]	Check for High volumes of access-denied errors from a specific identity.	TRUE
[h1.v2]	Checking for console login without MFA	TRUE
[h1.v3]	Checking for access-denied errors from creating an IAM user across multiple accounts.	FALSE

5.6.2 Hypothesis 2

This hypothesis focuses on ID: T1098.003. An adversary may add additional roles or permissions to a compromised cloud account to maintain persistent access and escalate privileges within an environment. This technique aligns with the MITRE ATT&CK framework under T1098.003 (Additional Cloud Roles), where adversaries manipulate IAM roles and permissions to ensure continued access and control over resources (MITRE, 2024). Figure 5.19 shows a description of T1098.003.

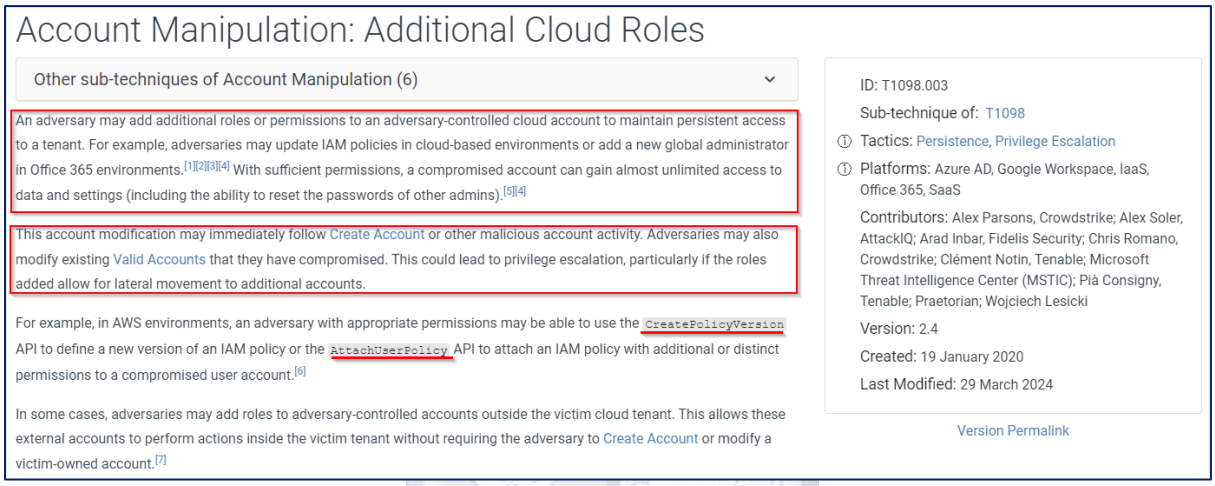


Figure 5.19: Hypothesis is based on ID: T1098.003

To validate this hypothesis, the following questions were considered and queries formulated as in Table 5.5; notably, for correlation validating the second hypothesis, evidence from findings of hypothesis 1 may heavily rely on.

i. [**h2.v1**] Are there roles or permissions that were modified? How do they deviate from normal operation patterns?
ii. [**h2.v2**] Is there evidence of privilege escalation after adding the new roles or permissions? Do any of these accounts gain access to sensitive resources or admin privileges
iii. [**h2.v4**] Are any unusual activities associated with the accounts with modified permission and roles?

Table 5.5: Validation analysis query for Hypothesis 2

```
import pandas as pd
import json
import os

# Specify the path to your CSV file
file_path = os.path.join(os.getcwd(), 'detected_cloudtrail_anomalies.csv')

iam_events_of_interest = [
    'AttachUserPolicy',
    'DetachUserPolicy',
    'PutRolePolicy',
    'DeleteRolePolicy',
    'CreatePolicyVersion',
    'SetDefaultPolicyVersion',
    'AttachGroupPolicy',
    'PutUserPolicy']
def process_csv(file_path):
    df = pd.read_csv(file_path)

    # Filter IAM events
    iam_events = df[df['eventSource'] == 'iam.amazonaws.com']
    filtered_df = iam_events[iam_events['eventName'].isin(iam_events_of_interest)]

    # Extract username and policy ARN from requestParameters
    filtered_df['username'] = filtered_df['requestParameters'].apply(lambda x:
json.loads(x).get('userName'))
    filtered_df['policyArn'] = filtered_df['requestParameters'].apply(lambda x:
json.loads(x).get('policyArn'))

    # Group by user identity, username, and policy ARN
    event_counts = filtered_df.groupby(['userIdentityArn', 'sourceIPAddress', 'username',
'policyArn']).agg(
```

```
            total_events=('userIdentityArn', 'count'),
            distinct_policies=('policyArn', 'nunique')
        ).reset_index()

        # Sort by total event count
        return event_counts.sort_values(by='total_events', ascending=False)

    if __name__ == "__main__":
        event_counts = process_csv(file_path)
        output_file_path = "iam_policy_counts.csv"
        event_counts.to_csv(output_file_path, index=False)
        print(f"Results saved to {output_file_path}")
```

An adversary may add additional roles or permissions to a compromised cloud account to maintain persistent access and escalate privileges within an environment. The results in Figure 5.20 validate the hypothesis questions in [h2.v1] and [h2.v2], showing role modifications and administrative policy assignments to escalate privileges. This investigation further explores the highlighted user activities for “tdir-workshop-nwolf-dev." “tdir-*workshop-sysdev*,” and “malicious_iam_user.” These accounts show suspicious behavior and may have been used for privilege escalation.

	eventName	userIdentityArn	sourceIPAddress	username	policyArn	total_events
1	AttachUserPolicy	arn:aws:iam::257394459111:user/cloudgoat	149.102.231.73	kerrigan	arn:aws:iam::257394459111:policy/cg-kerrigan-policy	1
2	AttachUserPolicy	arn:aws:iam::257394459111:user/cloudgoat	149.102.231.84	n_privesc_by_rollback_cgid47a6oqew6w	:y/cg-raynor-policy-iam_privesc_by_rollback_cgid47a6oqew6w	1
3	AttachUserPolicy	:iam::257394459111:user/tdir-workshop-nwolf-dev	54.204.116.47	tdir-workshop-sysdev	arn:aws:iam::aws:policy/AdministratorAccess	1
4	AttachUserPolicy	:iam::257394459111:user/tdir-workshop-nwolf-dev	54.204.116.47	tdir-workshop-nwolf-dev	arn:aws:iam::aws:policy/AdministratorAccess	1
5	AttachUserPolicy	arn:aws:iam::257394459111:user/stratus	109.81.160.122	malicious-iam-user	arn:aws:iam::aws:policy/AdministratorAccess	1
6	AttachUserPolicy	arn:aws:iam::257394459111:user/stratus	45.145.155.2	malicious-iam-user	arn:aws:iam::aws:policy/AdministratorAccess	1

Figure 5.170: Permission modification and Administrative Access policy assignment

The second hypothesis successfully answers two validation questions based on the findings and tabulated in Table 5.6: H2 Validation Results

Table 5.6: H2 Validation Results

	Hypothesis Validation	
[h2.v1]	Are there modified roles or permissions, and how do they deviate from standard operation patterns? Is the source IP of the role or permission modification known/regular?	TRUE
[h2.v2]	Is there evidence of the addition of administrative privileges	TRUE
[h2.v3]	Is there evidence of privilege escalation after adding the new roles or permissions? Do any of these accounts gain access to sensitive resources or admin privileges? Check account activities	FALSE

5.6.3 Hypothesis 3

This hypothesis focuses on MITRE ATT&CK ID T1136.003. An attacker may attempt to create an IAM user to maintain persistence in an environment they have breached. This technique aligns with the MITRE ATT&CK framework under T1136.003 (Cloud Account), where adversaries create cloud accounts to ensure continued access to victim systems without relying on persistent remote access tools (MITRE, 2024). Figure 5.21 shows a description of T1136.003.

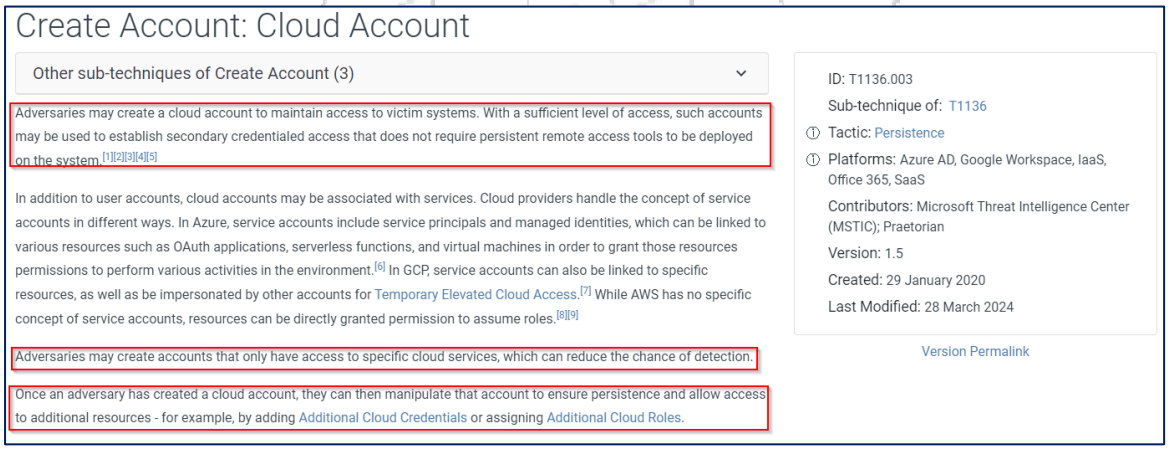


Figure 5.181: Hypothesis is created based on ID: T1136.003

As part of adversary persistence behavior, creating a cloud account enables unfettered access to the network, avoiding detection. The hypothesis was validated based on the questions outlined below.

i. [**h3.v1**] Quickly check for useridentityArn with multiple user creation, access key creation, and deletion activities.

Table 5.7: Validation analytic query for Hypothesis 3

```
import pandas as pd
import matplotlib.pyplot as plt

def analyze_user_activity(pdf):
    relevant_events = df[df['eventName'].isin(['CreateUser', 'CreateAccessKey', 'DeleteAccessKey', 'DeleteUser'])]
    user_activity_counts = relevant_events.groupby('userIdentityArn')['eventName'].value_counts().unstack(fill_value=0).reset_index()
    return user_activity_counts

if __name__ == "__main__":
    df = pd.read_csv("detected_cloudtrail_anomalies.csv")
    user_activity_counts = analyze_user_activity(df)

    user_activity_counts.set_index('userIdentityArn').plot(kind='bar', stacked=True, figsize=(15, 8))
    plt.title('User Activity Counts')
    plt.xlabel('User ID')
    plt.ylabel('Event Count')
    plt.xticks(rotation=45)
    plt.legend(title='Event Type')
    plt.tight_layout()
    plt.show()
```

Figure 5.21 shows the results of persistence and defense evasion techniques used by threat actors after gaining access to an AWS environment. From the analysis, the users “tdir-workshop-stiles-dev”, “tdir-workshop-nwolf-dev”, “tdir-workshop-sysdev,” and

“malicious_iam_user” stand out. The other accounts, including Cloudgoat, StrathAccTest, and Stratus, were configured profiles for these tests; hence, their higher count is expected.

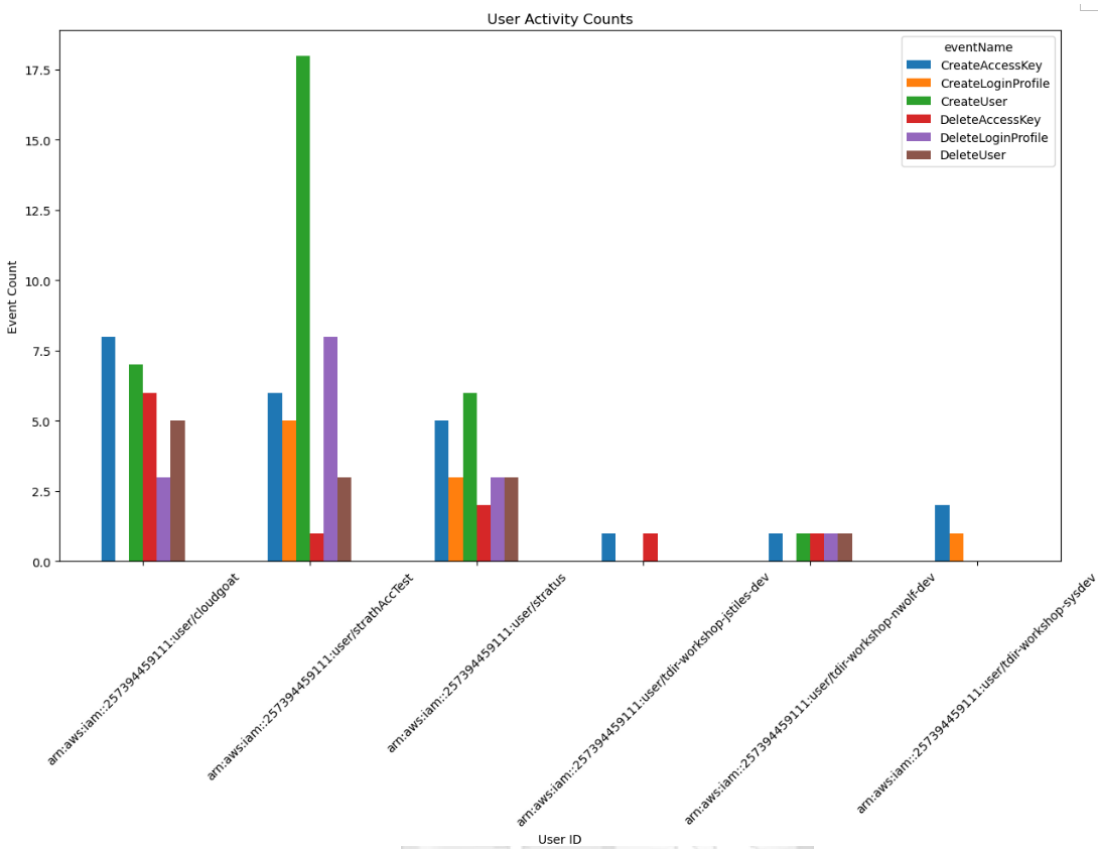


Figure 5.19: Persistence techniques

A further analysis was conducted on the creation of access keys by processing the detected_cloudtrail_anomalies.csv, explicitly focusing on IAM events related to access key creations. This involved aggregating the data by user and source, resulting in a comprehensive analysis outputted to a new CSV file, as illustrated in Figure 5.22. Such analysis is critical, given that the creation of access keys is a common tactic employed by threat actors to establish persistent access within the Amazon Web Services (AWS) environment. Long-lived access keys enable attackers to maintain unauthorized access even after an initial compromise, offering a significant advantage over simple user account creation by directly granting the ability to manipulate resources within AWS.

The findings from this analysis and insights from hypothesis 2 provide a reasonable basis to narrow the investigations to the users "tdir-workshop-nwolf-dev" and "tdir-workshop-system," considering the nature of their involvement activities. Combining the access key creation

patterns and associated user behaviours paints a great picture of these user actions, which is useful when creating a storyline for the investigation.

	userIdentityArn	sourceIPAddress	userName	accessKeyId	total__keys
1	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	admin_iam_privesc_by_key_rotation_cgid877rzxc0gd	AKIATX3PHYHTWD4LAYDK	1
2	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	admin_iam_privesc_by_key_rotation_cgid877rzxc0gd	AKIATX3PHYHTXTOCPPW4	1
3	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	developer_iam_privesc_by_key_rotation_cgid877rzxc0gd	AKIATX3PHYHT3T5EMDAV	1
4	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	manager_iam_privesc_by_key_rotation_cgid877rzxc0gd	AKIATX3PHYHTULNM2K65	1
5	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	raynor-iam_privesc_by_rollback_cgid0rzqqcdqc3	AKIATX3PHYHTRS6ZWE7O	1
6	arn:aws:iam::257394459111:user/cloudgoat	45.145.155.2	raynor-iam_privesc_by_rollback_cgidaesdybhl9d	AKIATX3PHYHT7WCMYGXQ	1
7	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-amansa-dev	AKIATX3PHYHTTFFZNXIV	1
8	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-mmajor-dev	AKIATX3PHYHTSP2ZYMGS	1
9	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-mmajor-dev	AKIATX3PHYHTVT7R6CWW	1
10	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-nwolf-dev	AKIATX3PHYHT4KFOHPDD	1
11	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-nwolf-dev	AKIATX3PHYHT7OQQSSBA	1
12	arn:aws:iam::257394459111:user/strathAccTest	109.81.160.83	tdir-workshop-sysdev	AKIATX3PHYHT7V47U4RT	1
13	arn:aws:iam::257394459111:user/stratus	45.145.155.2	stratus-red-team-backdoor-u-user	AKIATX3PHYHT3K43CBTX	1
14	arn:aws:iam::257394459111:user/tdir-workshop-jstiles-dev	54.226.202.95	tdir-workshop-rroe-dev	AKIATX3PHYHTUXALRR6F	1

Figure 5.23: Access Key Creations

While analyzing potential threat actor TTPs, the user account "dir-workshop-system" shows defense evasion tactics based on the analytics in Table 5.8.

Table 5.8: Defense Evasion validation analytics

```
import os
import pandas as pd
# Get the current working directory and construct the file path
file_path = os.path.join(os.getcwd(), "detected_cloudtrail_anomalies.csv")
# Read the CSV file
try:
    df = pd.read_csv(file_path)
Except Exception as e:
    print(f"Error reading the file: {e}")
    exit()
# Define defense evasion activities
defense_evasion_events = [
    'DeleteUser',
    'DeleteAccessKey',
    'DeleteLoginProfile',
    'DetachUserPolicy']
# Filter for defense evasion activities
defense_evasion_df = df[
    (df['eventSource'] == 'iam.amazonaws.com') &
    (df['eventName'].isin(defense_evasion_events))]
[ ['event name, 'eventTime', 'eventSource', 'sourceIPAddress',
     'userIdentityArn', 'errorMessage', 'errorCode',
     'requestParameters', 'userName', 'accessKeyId', 'userAgent']]
# Save the results to a CSV file
output_csv_path = os.path.join(os.getcwd(), "defense_evasion_activities_details.csv")
defense_evasion_df.to_csv(output_csv_path, index=False)
```

Figure 5.23 illustrates a series of actions that may be considered suspicious in an actual incident response investigation and would be helpful in building a timeline of findings. These include

the detachment of the Administrator policy, the user's Login Profile, and the deletion of the user account. Security teams can effectively correlate these events with other security telemetry, gaining valuable insights into a typical adversary's attack chain.

	username	userIdentityArn	eventName	ourceIPAddress	requestParameters	eventTime
1	stratus-red-team-nmfalu-nokeekqvzj	159111:user/strathAccTest	DeleteUser	109.81.160.83	{"userName": "stratus-red-team-nmfalu-nokeekqvzj"}	2024-12-02 17:53:43+00:00
2	stratus-red-team-nmfalu-nokeekqvzj	159111:user/strathAccTest	DeleteLoginProfile	109.81.160.83	{"userName": "stratus-red-team-nmfalu-nokeekqvzj"}	2024-12-02 17:53:42+00:00
3	tdir-workshop-rroe-dev	r/tdir-workshop-jstiles-dev	DeleteAccessKey	54.226.202.95	p-rroe-dev", "accessKeyId": "AKIATX3PHYHTUXALRR6F"}	2024-11-28 23:18:20+00:00
4	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteUser	45.145.155.2	e": "admin_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:03+00:00
5	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteUser	45.145.155.2	developer_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:03+00:00
6	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteUser	45.145.155.2	"manager_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:03+00:00
7	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteLoginProfile	45.145.155.2	developer_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:02+00:00
8	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteLoginProfile	45.145.155.2	e": "admin_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:01+00:00
9	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteLoginProfile	45.145.155.2	"manager_iam_privesc_by_key_rotation_cgid877rzxc0gd"}	2024-11-27 17:43:01+00:00
10	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteAccessKey	45.145.155.2	77rzxc0gd", "accessKeyId": "AKIATX3PHYHTWD4LAYDK"}	2024-11-27 17:42:56+00:00
11	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteAccessKey	45.145.155.2	7rzxc0gd", "accessKeyId": "AKIATX3PHYHTXTOCPPW4"}	2024-11-27 17:42:56+00:00
12	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DetachUserPolicy	45.145.155.2	olicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"}	2024-11-27 17:42:56+00:00
13	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteAccessKey	45.145.155.2	77rzxc0gd", "accessKeyId": "AKIATX3PHYHTULNM2K65"}	2024-11-27 17:42:56+00:00
14	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DeleteAccessKey	45.145.155.2	77rzxc0gd", "accessKeyId": "AKIATX3PHYHT3T5EMDAV"}	2024-11-27 17:42:56+00:00
15	sc_by_key_rotation_cgid877rzxc0gd	194459111:user/cloudgoat	DetachUserPolicy	45.145.155.2	olicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"}	2024-11-27 17:42:56+00:00
16	privesc_by_rollback_cgid0rzqqcdqc3	194459111:user/cloudgoat	DeleteUser	45.145.155.2	Name": "raynor-iam_privesc_by_rollback_cgid0rzqqcdqc3"}	2024-11-27 17:29:22+00:00
17	privesc_by_rollback_cgid0rzqqcdqc3	194459111:user/cloudgoat	DeleteAccessKey	45.145.155.2	zqqcdqc3", "accessKeyId": "AKIATX3PHYHTRS6ZWE7O"}	2024-11-27 17:29:21+00:00
18	privesc_by_rollback_cgid0rzqqcdqc3	194459111:user/cloudgoat	DetachUserPolicy	45.145.155.2	g-raynor-policy-iam_privesc_by_rollback_cgid0rzqqcdqc3"}	2024-11-27 17:29:21+00:00
19	stratus-red-team-login-profile-user	159111:user/strathAccTest	DeleteLoginProfile	45.145.155.15	{"userName": "stratus-red-team-login-profile-user"}	2024-11-26 08:35:19+00:00
20	stratus-red-team-login-profile-user	159111:user/strathAccTest	DeleteLoginProfile	45.145.155.15	{"userName": "stratus-red-team-login-profile-user"}	2024-11-26 08:35:08+00:00
21	tdir-workshop-sysdev	159111:user/strathAccTest	DeleteUser	109.81.160.83	{"userName": "tdir-workshop-sysdev"}	2024-11-25 07:59:11+00:00
22	tdir-workshop-sysdev	159111:user/strathAccTest	DeleteLoginProfile	109.81.160.83	{"userName": "tdir-workshop-sysdev"}	2024-11-25 07:58:39+00:00
23	tdir-workshop-sysdev	159111:user/strathAccTest	DetachUserPolicy	109.81.160.83	'policyArn": "arn:aws:iam::aws:policy/AdministratorAccess"}	2024-11-25 07:58:37+00:00

workshop-sysdev

Figure 5.20: Defense Evasion Actions

5.7 Threat Hunting Performance Validation

Validating a threat-hunting model's performance is paramount to guaranteeing its efficacy in identifying and mitigating potential threats within an organization's unique environment. Although threat hunting can be conducted manually, automated tools that utilize learned behavior to trigger detections significantly enhance the process. Here, we showcase how the model was validated.

5.7.1 Metrics Used for Evaluations

Outlier detection rate:

1. **Identify Actual Outliers**: Define criteria for an outlier in your CloudTrail logs. In the example, any event with the error code AccessDenied is considered an outlier.
2. **Count Actual Outliers**: Use the defined criteria to filter your processed data and count the outliers.
3. **Count Predicted Outliers**: Count how many anomalies were predicted by your model. In this case, instances labeled as -1 are considered anomalies.
4. **Calculate Accuracy Percentage**: Compute the accuracy of your model's predictions based on the counts obtained in steps 2 and 3. The accuracy is calculated as follows:
    - If there are actual outliers, use the following formula:
      Accuracy Percentage= (Predicted Outliers CountOutliers Counter) ×100Accuracy Percentage= (Outliers CounterPredicted Outliers Count) ×100
    - If there are no outliers, set accuracy to 0 to avoid division by zero.

```
# Step 7: Evaluate the model based on defined outlier rules for privilege escalation events
# Define a rule for actual outliers based on event types that indicate privilege escalation
actual_outlier_conditions = privilege_escalation_events['eventName'].isin([
    'CreateAccessKey',
    'CreatePolicyVersion',
    'UpdateAssumeRolePolicy',
    'AttachUserPolicy',
    'PutUserPermissionsBoundary'
])  # Add more relevant event types as needed

# Count actual outliers based on defined conditions
outliers_counter = actual_outlier_conditions.sum()

# Count predicted outliers by the model (-1 indicates an anomaly)
predicted_outliers_count = list(privilege_escalation_events['anomaly']).count(-1)

# Calculate accuracy percentage
accuracy_percentage = (predicted_outliers_count / outliers_counter * 100) if outliers_counter > 0 else 0

print("Actual number of outliers:", outliers_counter)
print("Predicted number of outliers by model:", predicted_outliers_count)
print("Accuracy percentage:", accuracy_percentage)
```

```
Actual number of outliers: 21
Predicted number of outliers by model: 4
Accuracy percentage: 19.047619047619047
```

Figure 5.221: Outlier detection code

Figure 5.25 above shows the actual outlier detection key components.

i. **Actual Outlier Detection**: The line filtering for "AccessDenied" events identifies critical security monitoring anomalies.

ii. **Predicted Outlier Count**: This sum counts the number of times the model flagged anomalies, which is essential for evaluating model performance.

iii. **Accuracy Calculation**: This part ensures you can derive a meaningful percentage reflecting how well your model performs against known outlier conditions.

The results from this computation are based solely on the model's predictions before applying it to real-time data. Therefore, they should not be interpreted as definitive performance metrics when deployed in live environments. Continuous monitoring and adjustment may be necessary to improve accuracy when working with dynamic datasets like AWS CloudTrail logs, which can vary significantly over time due to changes in user behavior and system configurations. This approach allowed the researcher to assess the effectiveness of the anomaly detection model and make informed decisions about further tuning or adjustments needed for real-time applications.

Isolation Forest Model Evaluation

Cross-validation evaluated the isolation forest model's performance. It helped assess the model's robustness by providing a reliable estimate of its performance, reducing overfitting, and identifying data-related issues. The average score from the cross-validation is 0.5056628817862147, as captured in Figure 5:26.

Cross Validation

```
from sklearn.model_selection import cross_val_predict
### using cross validation to check consistency of model prediction across folds
scores = cross_val_predict(IsolationForest(random_state=42, contamination=0.1, n_estimators=250, max_samples ='auto'), features, cv=5, method="pre
```

```
print(f"Cross-Validation Scores: {scores}")
print(f"Average Score: {scores.mean()}")
```

```
Cross-Validation Scores: [ 1 -1 -1 ...  1  1  1]
Average Score: 0.5056628817862147
```

Figure 5.226: Isolation Forest model validation

Figure 5.27 shows the predicted class labels for each data point in each cross-validation fold. Each prediction corresponds to whether the model classified that point as an anomaly (-1) or standard (1). With an average score of (0.55), which is the mean accuracy score for the model across the folds

```
# Fit the model and get predictions
features_df['anomaly_score'] = model.fit_predict(features)

# Get the decision function scores (anomaly scores)
features_df['decision_function'] = model.decision_function(features)

# Display the updated dataframe with anomaly labels and scores
print(features_df[['anomaly_score', 'decision_function']].head())
```

```
   anomaly_score  decision_function
0              1           0.054983
1             -1          -0.012358
2              1           0.008432
3             -1          -0.001089
4             -1          -0.014649
```

Figure 5.27: Predicted Class labels average score

5.8 Conclusion

This chapter examines the deployment of the threat-hunting model, detailing both offensive and defensive security methodologies. The model is validated, and its effectiveness in refining hypotheses and enhancing threat hunting is evaluated by further analyzing the model output “detected_cloudtrail_anomalies.csv" results.” Three hypotheses are formulated based on MITRE ATT&CK framework and validation questions are used as a guiding principle in the validation of these hypothesis. The validation question is further augmented by analytic queries on the output of the TH model. While the model may produce some false positives and biases, it acts a proof value of threat hunting in threat hunting and fulfills the study's objectives, demonstrating its ability to predict outlier activities which a can further be analyzed to identify potential threats activities in given aws environment.

Chapter 6: Discussion

6.1 Overview

This chapter analyses the key findings from the study. The threat-hunting platform was validated to see whether it met the research objectives.

6.2 Reviewing Threat-Hunting Approaches in the Cloud.

The initial objective was achieved through a comprehensive review of the literature. In particular, prior research indicates that misconfigurations in AWS Cloud, particularly those stemming from Identity and Access Management (IAM), represent a significant threat vector. The study further detailed typical threat scenarios exploited by threat actors to manipulate IAM, with a particular emphasis on privilege escalation and persistence, and considers this aspect as one of the primary focuses for any threat-hunting effort. Although previous studies demonstrate a trend indicating reliance on machine learning (ML) and artificial intelligence (AI) and underscore the importance of incorporating offensive and counter-offensive measures, most of these efforts have only been concentrated on on-premises environments. This is despite the emergence of cloud-conscious threat actors. There exists a notable deficiency in understanding attackers' behaviors within cloud environments, particularly in the context of the emergence of these cloud-oriented threat actors. Consequently, the study identifies a gap in implementing threat hunting in the cloud while still relying on ML/AI and integrating Cyber Threat Intelligence (CTI) with knowledge base frameworks, such as MITRE ATT&CK.

6.3 Adversary Emulation for Achieving Threat Hunting in the Cloud.

The second objective was to review adversary emulation methods that can be used for threat hunting in the cloud. This objective was also achieved through a systematic literature review. The literature indicates that threat emulation in cloud environments is a strategic approach to operationalizing threat-hunting capabilities by mimicking threat actors and APT groups, similar to a purple teaming effort. While existing tools for adversaries are scarce, very few focus on the cloud environment. However, the cloud would still benefit from adversary emulation by providing a realistic way to evaluate security controls and enhance detections and response capabilities following threat-hunting actions like detection engineering.

The study also illustrates that adversary emulation in the cloud can target specific threat vectors, such as IAM misconfigurations. Therefore, developing a comprehensive emulation plan is essential and should be based on an understanding of the APT groups that target the

industry of operation, which requires gathering relevant CTI as a guiding point. Ideally, the emulation plan should always include all post-compromise activities.

While there are tools for adversary emulation in the cloud, they are not robust enough to cover all attacker tactics. Therefore, it is important to review notable attack paths, especially those targeting IAM misconfigurations, such as privilege escalation, and to create a test library that includes both manual and automated tests. The identified tools, such as the Stratus Red Team, enable the execution of diverse cloud-native attacks across platforms, including AWS and Azure. From the review, it is clear that cloud attack emulation can be integrated with detection engineering for continuous testing, which may help an organization adapt to the emergence of new threats.

6.4 Developing and Testing the TH Platform.

The third objective was to design, develop, and test a platform for operationalizing threat hunting in the AWS cloud. This is achieved through a design science approach that leverages an experimental methodology divided into offensive and counter-offensive phases. A threat-hunting model utilizing machine learning powered by analytics is employed; it features an unsupervised learning model based on the isolation forest algorithm, which is suitable for outlier detection. The data requirements for this research include cyber threat intelligence reports that provide the necessary data for CTI on APT groups or threat actor groups targeting the cloud, as understanding their operations is critical for both offensive and counter-offensive aspects of the research. Another important dataset for this research was the AWS Cloud log public dataset, which was used to train the TH model.

A comprehensive design structure is provided to guide the implementation of the research, including the design of the threat-hunting platform, the platform flowchart, the design of use cases, and the model sequence diagram. Subsequently, an analysis of selected vendor threat intelligence feeds was conducted to gain insight into the tactics of cloud-conscious threat actors. Test stories were developed as part of the offensive phase, and a threat-hunting heatmap was constructed to guide the counter-offensive phase of the research. Finally, three hypotheses were formulated to operationalize the hunt, with the objective of either rejecting or confirming the hypothesis questions.

6.5 Validating the Performance of the Threat-Hunting Platform

The final objective of this research endeavor was to demonstrate the implementation of the threat-hunting platform by validating its performance through a comprehensive overview of

the isolation-forest-based threat model. This model captured the algorithm and how features were encoded using a label encoder algorithm. In the context of this research, the offensive phase involved tests divided into automated and manual approaches. The primary advantage of automating attacks lies in their flexibility, enabling assessments to be regenerated multiple times. This facilitates a more comprehensive understanding of the resulting log data for threat detection and hunting while enhancing model performance. By automating tests within a typical organization, we ensure that security vulnerabilities do not re-emerge during significant changes.

The counter-offensive part of the research commenced by mapping the hypotheses developed in Chapter 4 into the MITRE ATT&CK framework and creating validation questions for each hypothesis. Four hypotheses were formulated as part of this project, establishing a systematic testing approach.

Subsequently, the model validation entailed an analysis of outlier detection and cross-validation of the isolation forest model. While the cross-validation graph suggests a typical data distribution, it is crucial to recognize potential biases existing in the training dataset. Given the dynamic nature of cyber threats and AWS environments, the age of the dataset may hinder its ability to represent emerging attack patterns or new event types accurately. This limitation could lead to a decrease in model performance when applied to real-time CloudTrail logs. Further analysis was conducted on the resulting model output of “detected_cloudtrail_anomalies.csv." Although the provided model offers a basic framework for analyzing CloudTrail logs, it has limitations in detecting subtle anomalies and emerging threats, a shortcoming driven by time constraints.

Enriched Feature Engineering could improve effectiveness by making several enhancements. These include incorporating contextual features such as user roles, resource types, and geographic location to provide a richer context for anomaly detection and analyzing user behavior patterns to identify deviations from regular activity.

Beginning with hypothesis 1, the threat-hunting model classifies events yielding resulting error messages as anomalous. This phase's threat-hunting effort aimed to correlate further whether these were isolated out-of-context events or part of a larger attack chain. However, the model could not validate instances where an IAM user logged in with MFA due to the limitations of the feature events present in the training model.

The second hypothesis was also confirmed, as the model successfully classified role modifications, particularly assigning administrative policies that grant an IAM entity elevated privileges. This allows for a broader range of actions, including exfiltration and lateral movement. Nonetheless, challenges arose as this aspect exhibited potential bias, which necessitates a comprehensive understanding of the entire attack context—correlating diverse events to effectively assert that an administrative policy assignment, for example, could be linked to a compromised IAM user leveraging the account to escalate privileges.

The model's ability to classify user creation, access key creation, and subsequent deletion events validated the third hypothesis. To further refine the analysis and enhance the model's accuracy, incorporating additional contextual data, such as geolocation and user agents, could foster a correlation between these events and other malicious activities while minimizing false positives. Continuous tuning and refinement of the model are essential for bolstering its performance and improving the detection of emerging threats.

6.6 Achievement of Objectives

This section summarized how each research objective was accomplished:

i. Objective 1: Reviewed Threat-Hunting Approaches in the Cloud
Achieved through a literature review identifying IAM misconfigurations as a key threat and highlighting the need for ML/AI and CTI integration in cloud threat hunting.

ii. Objective 2: Evaluated Adversary Emulation Methods for Cloud Threat Hunting
Achieved by systematically reviewing adversary emulation methods and demonstrating their application in cloud environments, focusing on IAM attack vectors and tools like Stratus Red Team.

iii. Objective 3: Designed, Developed, and Tested a Threat-Hunting Platform for AWS.
Achieved by developing a machine-learning-based platform using the isolation forest algorithm, informed by CTI and AWS CloudTrail logs, guided by a structured design and implementation process.

iv. Objective 4: Validated the Performance of the Threat-Hunting Platform
Achieved through comprehensive automated and manual testing, mapping hypotheses to the MITRE ATT&CK framework, and evaluating the model’s ability to detect anomalies.

6.7 Study Contributions

I. Addresses a knowledge gap in cloud threat hunting

   a. Provided a focused review of threat-hunting approaches in cloud environments, especially AWS, highlighting IAM misconfigurations as a significant risk. It also identified the lack of cloud-specific methodologies and the need to integrate ML/AI and CTI in cloud threat hunting.

   b. Incorporated offensive (adversary emulation) and counter-offensive (detection engineering) phases, guided by real-world CTI and public threat datasets for developing the model.

II. Advancing adversary emulation methods for the cloud

   a. Systematically reviewed and demonstrated the application of adversary emulation in cloud settings, using tools like Stratus Red Team.

   b. Emphasized the importance of developing comprehensive emulation plans based on relevant CTI, combining automated and manual testing.

III. Promotes creation of custom detection models

   a. Encouraged and demonstrated the development of customized threat-hunting models tailored to each enterprise's unique characteristics and needs. This supports the alignment of detection strategies with specific organizational environments (e.g., cloud, OT, or on-premises).

6.8 Recommendations

In light of the findings, several recommendations are proposed: In today's rapidly evolving landscape of cybersecurity threats, organizations must adopt innovative strategies to safeguard their systems. One promising approach is leveraging Large Language Models (LLMs) for advanced threat modeling. By employing these powerful models, organizations can generate realistic and sophisticated attack scenarios, enabling them to proactively identify and address potential vulnerabilities before they can be exploited.

However, the uniqueness of each organization's environment necessitates using a customized model. Whether an enterprise operates with operational technology (O.T.), cloud infrastructure, or solely on-premises solutions, having a tailored model accelerates the

detection of threats and deepens the analysis of those threats, ensuring that specific vulnerabilities are properly addressed.

Organizations should integrate LLMs into their Security Orchestration, Automation, and Response (SOAR) platforms to further enhance their security posture. This integration allows for the automation of incident response workflows, accelerates threat detection processes, and ultimately improves overall security operations. By combining LLMs' innovative capabilities with customized approaches and robust SOAR strategies, organizations can significantly bolster their defenses against the ever-increasing tide of cyber threats.

6.9 Future of Work

In light of the findings, the following future of work is proposed

1 Integrating Advanced LLM-Powered Threat Hunting will revolutionize cybersecurity strategies in the evolving work landscape. Organizations will harness sophisticated LLM-based techniques to enhance anomaly detection, refine threat intelligence analysis, and streamline incident response. As threats grow increasingly complex, the focus will shift towards supporting advanced, non-atomic attacks—research will prioritize methods that effectively correlate multifaceted, multi-stage attacks. This innovative approach will incorporate advanced techniques such as behavioral analytics and machine learning, allowing for the identification of intricate attack patterns that will be previously undetectable.

2 To complement these technological advancements, developing hybrid approaches will become paramount. By combining human expertise with AI capabilities, organizations will achieve optimal threat detection and response results, paving the way for a safer work environment. This synergy will ensure that while machines analyze vast datasets, human intuition and experience will guide decision-making processes in real-time.

3 A robust framework for assessing detection quality will also be essential in this future workforce. This framework will consider critical factors such as Threat Alignment and Coverage, Detection Integrity, and Utility. By employing such a structure, organizations will continually gauge the effectiveness of their cybersecurity measures, adapting to new threats as they arise and ultimately fostering a resilient and proactive work environment.

4 As we look towards the future of work, it is clear that the collaboration between advanced technology and human insight will define our success in navigating the complexities of

cybersecurity, ensuring the safety of our digital ecosystems and the integrity of our collective workforce.

6.10 Conclusion

The study on adversary emulation methods for threat hunting in cloud environments underscores the critical need for organizations to adapt their cybersecurity approaches to address the unique challenges posed by cloud-based threats. Research findings reveal that while traditional threat-hunting techniques are well-established, there remains a significant knowledge gap regarding the behaviors of attackers aware of the risks associated with cloud computing. This gap necessitates the development of tailored methodologies that leverage advanced technologies like machine learning (ML) and artificial intelligence (AI) to enhance threat detection and response capabilities.

The systematic literature review indicates that adversary emulation is a strategic framework for proactively assessing security postures in cloud environments. Tools such as Stratus Red Team enable the simulation of various cloud-native attacks, providing organizations with a deeper understanding of potential vulnerabilities and the effectiveness of current security measures. Integrating adversary emulation with detection engineering facilitates ongoing testing, allowing organizations to adapt to the evolving threat landscape.

Furthermore, the development and evaluation of a comprehensive threat-hunting platform demonstrate the practical implementation of these methods. The platform's design, thorough testing, and hypothesis validation yield valuable insights into cloud security dynamics and underscore the significance of contextual data in enhancing anomaly detection. Finally the study provides relatable future work, and a large language model will be used to develop even better threat models by also providing recommendations

In conclusion, this research emphasizes the necessity for organizations to implement threat hunting through adversary emulation, fostering a proactive cybersecurity stance capable of effectively addressing emerging threats in cloud environments. Such strategies must be continuously refined and adapted to maintain robust defenses against increasingly sophisticated cyber threats.

References

Ajmal, A. B., Shah, M. A., Maple, C., Asghar, M. N., & Islam, S. U. (2021). Offensive security: Towards proactive threat hunting via adversary emulation. *IEEE Access*, 9, 126023–126033. https://doi.org/10.1109/ACCESS.2021.3104260

Alam, M. T., Bhusal, D., Park, Y., & Rastogi, N. (2023, October). Looking beyond IoCs: Automatically extracting attack patterns from external CTI. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses* (pp. 92–108).

Alavizadeh, H., Alavizadeh, H., & Jang-Jaccard, J. (2020). Cyber situation awareness monitoring and proactive response for enterprises on the cloud. *https://doi.org/10.1109/trustcom50675.2020.00171*

Al-Karaki, J., Gawanmeh, A., Almalkawi, I., & Alfandi, O. (2020). Probabilistic analysis of security attacks in cloud environment using hidden Markov models. *Transactions on Emerging Telecommunications Technologies*, 4(33). https://doi.org/10.1002/ett.3915

Araujo, F., Kirat, D., Shu, X., Taylor, T., & Jang, J. (2021). Evidential cyber threat hunting. *http://arxiv.org/abs/2104.10319*

Bakker, D. R. (2022). Autonomous emulation of adversary procedures in the (pre-)compromise domain.

Chamberlain, C. (2021, March 3). Detecting threats in AWS CloudTrail logs using machine learning. *Elastic*. https://www.elastic.co/blog/detecting-threats-in-aws-cloudtrail-logs-using-machine-learning (Accessed January 2, 2024).

Chen, C. K., Lin, S. C., Huang, S. C., Chu, Y. T., Lei, C. L., & Huang, C. Y. (2022). Building machine learning-based threat-hunting system from scratch. *Digital Threats: Research and Practice*, 3(3). https://doi.org/10.1145/3491260

CrowdStrike. (2023). The rise of the cloud-conscious adversary: 2023 cloud risk report.

Daszczyszak, R., Ellis, D., Luke, S., & Whitley, S. (2019). *Sponsor: USCYBERCOM TTP-Based Hunting*.

Detecting the Unknown: A Guide to Threat Hunting. (2019).

Diogenes, Y., et al. (2023). *Exam Ref SC-100 Microsoft Cybersecurity Architect*. Microsoft Press.

Gao, P., Shao, F., Liu, X., Xiao, X., Qin, Z., Xu, F., Mittal, P., Kulkarni, S. R., & Song, D. (2021). Enabling efficient cyber threat hunting with cyber threat intelligence. *Proceedings - International Conference on Data Engineering*, *2021-April*, 193–204. https://doi.org/10.1109/ICDE51399.2021.00024

Goundar, S. (n.d.). *Chapter 3-Research Methodology and Research Method*. https://www.researchgate.net/publication/333015026

Gunter, D., & Seitz, M. (2021). *A Practical Model for Conducting Cyber Threat Hunting*.

Hillier, C., & Karroubi, T. (2022). *Turning the Hunted into the Hunter via Threat Hunting: Life Cycle, Ecosystem, Challenges and the Great Promise of AI*. http://arxiv.org/abs/2204.11076

Holm, H., & Sommestad, T. (n.d.). *Highlights Realistic and Balanced Automated Threat Emulation Realistic and Balanced Automated Threat Emulation*. https://github.com/guardicore/monkey

Howley, C. (2023). Gartner forecasts worldwide public cloud end-user spending to reach $679 billion in 2024. Gartner. Retrieved from https://www.gartner.com/en/newsroom/press-releases/11-13-2023-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-679-billion-in-20240 (Accessed November 24, 2023).

Intrusion Detection Guide. (n.d.).

Jiang, H., Nagra, J., & Ahammad, P. (2016). *SoK: Applying Machine Learning in Security - A Survey*. http://arxiv.org/abs/1611.03186

Kharma, M., & Taweel, A. (2022, December). Threat modeling in cloud computing—a literature review. In International Conference on Ubiquitous Security (pp. 279–291). Springer Nature Singapore.

Khorshed, M. T., Ali, A. S., & Wasimi, S. A. (2012). A survey on gaps, threat remediation challenges, and some thoughts for proactive attack detection in cloud computing. Future Generation Computer Systems, 28(6), 833–851.

Lee, R. M., & Lee, R. T. (2018). SANS 2018 threat hunting survey results. SANS Institute Reading Room.

Mahboubi, A., Luong, K., Aboutorab, H., Bui, H. T., Jarrad, G., Bahutair, M., Camtepe, S., Pogrebna, G., Ahmed, E., Barry, B., & Gately, H. (2024). Evolving techniques in cyber threat hunting: A systematic review. In *Journal of Network and Computer Applications* (Vol. 232). Academic Press. https://doi.org/10.1016/j.jnca.2024.104004

Mandiant, (2022). A TIERED FRAMEWORK FOR CYBER THREAT LEVELS

Mavroeidis, V., & Jøsang, A. (2018). Data-driven threat hunting using sysmon. *ACM International Conference Proceeding Series*, 82–88. https://doi.org/10.1145/3199478.3199490

MITRE ATT&CK in Amazon Web Services (AWS): A defender's cheat sheet. (n.d.). www.expel.io

MITRE. (2013, July 21). MITRE ATTaCK®. *Attack*. https://attack.MITRE.org/

MITRE. (2019). Detecting the unknown: A guide to threat hunting.

Network Threat Hunter Training. (n.d.). https://www.antisyphontraining.com/advanced-network-threat-hunting-w-chris-brenton/

Nour, B., Pourzandi, M., & Debbabi, M. (2023). A Survey on Threat Hunting in Enterprise Networks. *IEEE Communications Surveys and Tutorials*, *25*(4), 2299–2324. https://doi.org/10.1109/COMST.2023.3299519

Peris, C., Pillai, B., & Kudratti, A. (n.d.). Threat hunting in the cloud.

Piper, S. (2020). Public dataset of CloudTrail logs from flaws.cloud, Summit route – public dataset of CloudTrail. Retrieved from [URL_6]

Raju, A. D., Abualhaol, I. Y., Giagone, R. S., Zhou, Y., & Huang, S. (2021). A survey on cross-architectural IoT malware threat hunting. *IEEE Access*, 9, 91686–91709. https://doi.org/[DOI_7

Razi, P. (n.d.). *MITRE ATTACKS DETECTION RULES PART1 The MITRE ATT&CK Alerts For log point*.

Shahar, A. (2022, April 22). Strategies for the cloud threat hunter. *Devm*. Retrieved from https://devm.io/cloud/cloud-threat-hunter-177233

Shan, A., & Myeong, S. (2024). Proactive Threat Hunting in Critical Infrastructure Protection through Hybrid Machine Learning Algorithm Application. *Sensors*, *24*(15). https://doi.org/10.3390/s24154888

Sheikhi, S., & Kostakos, P. (2023). Cyber threat hunting using unsupervised federated learning and adversary emulation. *Proceedings of the 2023 IEEE International Conference on Cyber Security and Resilience, CSR 2023*, 315–320. https://doi.org/10.1109/CSR57506.2023.10224990

Splunk. (n.d.). *The PEAK Threat Hunting Framework*.

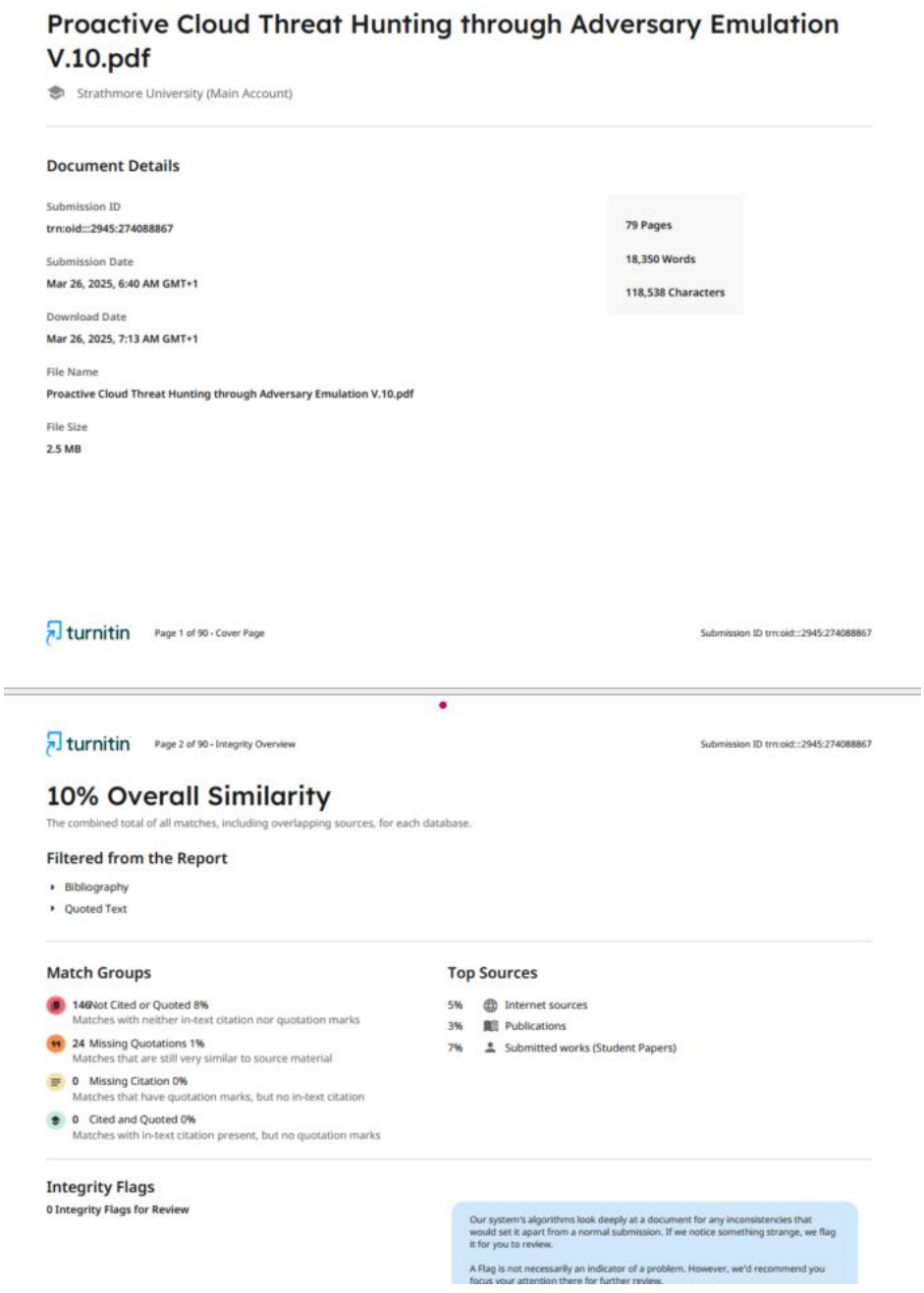
Steen, A. B., & Khasuntsev, N. (2021). *Automatic Detection of Misconfigurations of AWS Identity and Access Management Policies*.

Tafani-Dereeper, C. (n.d.). *Purple Teaming & Adversary Emulation in the Cloud with Stratus Red Team*.

Van Buggenhout, E. (2022). *The Advantages of Going Purple: How BAS Works and Why It Matters*. https://www.bls.gov/ooh/computer-and-information-technology/information-security-analysts.htm

Appendices

Appendix A: Similarity Score



Proactive Cloud Threat Hunting through Adversary Emulation V.10.pdf

Strathmore University (Main Account)

Document Details

Submission ID
trn:oid:::2945:274088867

Submission Date
Mar 26, 2025, 6:40 AM GMT+1

Download Date
Mar 26, 2025, 7:13 AM GMT+1

File Name
Proactive Cloud Threat Hunting through Adversary Emulation V.10.pdf

File Size
2.5 MB

79 Pages

18,350 Words

118,538 Characters

turnitin Page 1 of 90 - Cover Page Submission ID trn:oid:::2945:274088867

turnitin Page 2 of 90 - Integrity Overview Submission ID trn:oid:::2945:274088867

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

146 Not Cited or Quoted 8%
Matches with neither in-text citation nor quotation marks

24 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
3% Publications
7% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Appendix B: Ethical Clearance Letter

Strathmore UNIVERSITY

21st March 2025

Mr Bunde Collins,
collins.bunde@strathmore.edu

Dear Mr Bunde,

RE: Proactive Cloud Threat Hunting Through Adversary Emulation

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2566/25.** The approval period is from **21st March 2025 to 20th March 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC