

Detection of Attacker Activity in Cloud Environments

Bc. Jan Blažek

Master's thesis
2025



Tomas Bata University in Zlín
Faculty of Applied Informatics

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Bc. Jan Blažek
Osobní číslo: A23400
Studijní program: N0613A140022 Informační technologie
Specializace: Kybernetická bezpečnost
Forma studia: Kombinovaná
Téma práce: Detekce aktivity útočníků v cloudových prostředích
Téma práce anglicky: Detection of Attacker Activity in Cloud Environments

Zásady pro vypracování

- Zpracujte přehled zabezpečení cloudových prostředí.
- Provedte detailní analýzu Amazon Web Services (AWS) se zaměřením na řešenou problematiku.
- Popište techniky, taktiky a postupy (TTP) útočníků v cloudových prostředích.
- Vytvořte detekční vzory pro vybrané techniky.
- Navržené vzory ověřte v testovacím prostředí.

Forma zpracování diplomové práce: **tištěná/elektronická**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

1. RAMAKRISHNAN, Ganesh a Mansoor HAQANEE. *Cloud Forensics Demystified: Decoding cloud investigation complexities for digital forensic professionals*. Packt Publishing, 2024. ISBN 1800564414.
2. *Practical Threat Detection Engineering: A hands-on guide to planning, developing, and validating detection capabilities*. Packt Publishing, 2023. ISBN 1801076715.
3. NETO, Marcello Zillo; SANTANA, Gustavo Alessandro Andrade; SAPATA, Fernando; MUNOZ, Mauricio; MORAES, Alexandre Matos da Silva Pires de et al. *AWS Certified Security Study Guide*. Indianapolis: Sybex, 2021. ISBN 9781119658856.
4. LAT, Joshua Arvin. *MACHINE LEARNING ENGINEERING ON AWS*. S.I.: PACKT PUBLISHING LIMITED, 2022. ISBN 9781803231389.
5. GUPTA, C. P. a GOYAL, K. K. *Cybersecurity*. 2020. ISBN 9781683924968.

Vedoucí diplomové práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **27. října 2024**
Termín odevzdání diplomové práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA DIPLOMOVÉ PRÁCE

Beru na vědomí, že

- odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení diplomové práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – diplomovou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na diplomové práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Diplomová práce se zaměřuje na detekci aktivit útočníků v cloudových prostředích, se zvláštním důrazem na cloudovou platformu Amazon Web Services (AWS). Práce popisuje základní charakteristiky cloudových prostředí, jejich typy, infrastrukturu, služby a produkty poskytované cloudovými platformami a jejich vzájemné propojení. Dále zkoumá techniky, taktiky a postupy (TTP), které mohou být útočníky využity k provádění škodlivých aktivit v těchto prostředích. Součástí práce je návrh a implementace detekčních vzorů specifických technik v běžně používaných detekčních jazycích, jako je Sigma. Výsledkem je vytvoření nástrojů a metodologie, které mohou sloužit k efektivnější ochraně cloudových prostředí před kybernetickými hrozbami.

Klíčová slova: Cloudová bezpečnost, Amazon Web Services, detekce útočníků, TTP, Sigma

ABSTRACT

The thesis focuses on detecting adversary activities in cloud environments, with a particular emphasis on the Amazon Web Services (AWS) cloud platform. It describes the fundamental characteristics of cloud environments, their types, infrastructure, services, and products provided by cloud platforms, as well as their interconnections. Furthermore, the thesis examines techniques, tactics, and procedures (TTP) that adversaries may use to conduct malicious activities in these environments. The work includes the design and implementation of detection patterns for specific techniques using commonly employed detection languages such as Sigma. The result is the development of tools and methodologies that can enhance the protection of cloud environments against cyber threats.

Keywords: Cloud security, Amazon Web Services, adversary detection, TTP, Sigma

I would like to express my sincere gratitude to Ing. David Maláník, Ph.D. for his expert guidance and consistent support throughout my research and writing. His insights into cloud security and detection engineering provided a strong foundation for my work, and his constructive feedback helped shape the final form of this thesis.

TABLE OF CONTENTS

INTRODUCTION	13
I THEORETICAL PART	15
1 INTRODUCTION TO CLOUD COMPUTING.....	16
1.1 ADVANTAGES OF CLOUD COMPUTING	16
1.2 CLOUD DISADVANTAGES.....	18
1.3 TYPES OF CLOUD SERVICES	19
1.3.1 Infrastructure as a Service.....	20
1.3.2 Platform as a Service	20
1.3.3 Software as a Service.....	21
2 AN OVERVIEW OF CLOUD SECURITY	22
2.1 THE SHARED RESPONSIBILITY MODEL	22
2.2 KEY THREATS IN CLOUD SECURITY	23
2.2.1 Misconfigurations	23
2.2.2 Data Breaches.....	23
2.2.3 Distributed Denial of Service (DDoS) Attacks	24
2.2.4 Insider Threats	24
2.2.5 Credential Theft	24
2.3 STRENGTHENING CLOUD SECURITY POSTURE.....	25
2.3.1 Proactive Security Measures.....	25
2.3.2 Reactive Security Measures.....	25
2.4 KEY PRINCIPLES OF CLOUD SECURITY	25
2.5 CONCLUSION	26
3 AMAZON WEB SERVICES SECURITY CONTROLS.....	27
3.1 IDENTITY & ACCESS MANAGEMENT	27
3.1.1 AWS Identity and Access Management.....	27
3.1.2 AWS IAM Identity Center	28
3.1.3 AWS Secrets Manager	30
3.2 NETWORK SECURITY.....	31

3.2.1	AWS Web Application Firewall	31
3.2.2	Amazon VPC Security Groups.....	32
3.3	SECURITY MONITORING & THREAT DETECTION.....	33
3.3.1	AWS Security Hub.....	33
3.3.2	Amazon GuardDuty	35
3.3.3	AWS CloudTrail.....	37
4	EVALUATION OF AWS SECURITY SERVICES.....	40
4.1	IAM SERVICES	42
4.1.1	Confidentiality	42
4.1.2	Integrity	43
4.1.3	Availability	43
4.1.4	Compliance	44
4.1.5	Cost	45
4.1.6	Scalability.....	46
4.1.7	Final Score.....	47
4.2	INCIDENT RESPONSE SERVICES.....	47
4.2.1	Confidentiality	47
4.2.2	Integrity	48
4.2.3	Availability	49
4.2.4	Compliance	49
4.2.5	Cost	50
4.2.6	Scalability.....	51
4.2.7	Final Score.....	51
4.3	COMPLIANCE SERVICES	52
4.3.1	Confidentiality	52
4.3.2	Integrity	53
4.3.3	Availability	53
4.3.4	Compliance	54
4.3.5	Cost	55
4.3.6	Scalability.....	56
4.3.7	Final Score.....	56

5	DETECTION ENGINEERING CONCEPTS	58
5.1	SECURITY INFORMATION AND EVENT MANAGEMENT.....	58
5.2	PYRAMID OF PAIN.....	59
5.3	TACTICS, TECHNIQUES AND PROCEDURES	60
5.3.1	Valid Accounts.....	60
5.3.2	Crete Account.....	61
5.3.3	Event Driven Execution	62
5.3.4	Modify Authentication Process	64
5.3.5	Remote Services.....	65
5.3.6	Credential Access	66
5.4	SIGMA.....	67
II	PRACTICAL PART.....	68
6	ADVERSARY ACTIVITY DETECTION	69
6.1	RULE STRUCTURE	69
7	DETECTION RULES.....	71
7.1	SHARED RULES	71
7.1.1	IAM Persistence	71
7.1.2	Successful AWS Console Sign-in	72
7.1.3	Automated Execution	73
7.2	ROOT SIGN-IN WITHOUT MULTI-FACTOR AUTHENTICATION	73
7.3	ENUMERATE-IAM TOOL	75
7.3.1	Deployment	77
7.4	FEDERATION TOKEN ABUSE	78
7.4.1	Detection Rules	79
7.4.2	Excluding Legitimate Activity	81
7.5	SINGLE SIGN-ON CREDENTIAL THEFT	82
7.5.1	SSO Authentication Workflow	82
7.5.2	AWS SSO and AWS CLI.....	83
7.5.3	Detection Strategies	83
7.6	LAMBDA EXECUTING BACKDOOR CODE.....	86

7.6.1	Observed Anomalies	86
7.6.2	Detection	87
7.7	TRUFFLEHOG ABUSE	90
7.7.1	Detection	91
7.8	GOLDEN SECURITY ASSERTION MARKUP LANGUAGE ATTACK	92
7.9	PACU	94
7.10	UNRESTRICTED ACCESS TO SENSITIVE PORTS	97
7.11	HIGH VOLUME OF UNUSUAL API CALLS	98
7.11.1	Advantages of Machine Learning for Anomaly Detection	99
CONCLUSION		101
REFERENCES		103
LIST OF FIGURES		109
LIST OF LISTINGS		110
LIST OF ABBREVIATIONS		112
LIST OF APPENDICES		114

INTRODUCTION

Cloud computing is also now a fundamental component of contemporary information technology infrastructures, providing organizations with agile resources and economic solutions. Such swift uptake of cloud services has initiated new security issues that need careful consideration, particularly in threat identification and mitigation contexts. Among the prominent cloud service providers, Amazon Web Services (AWS) is notable for its broad array of services, vast global footprint, and strong underlying infrastructure. Yet, precisely because of the broad set of AWS services, it is a complicated environment in which attackers may take advantage of misconfigurations, vulnerabilities, or inadequate monitoring practices. Thus, efficient detection mechanisms need to entail not just conventional security controls but also specialized methods and techniques designed for cloud-specific threats.

The current thesis aims to explore the intricacies involved with AWS security and provide detection patterns that respond to relevant attacker techniques. The initial chapters will provide an introduction of basic concepts around cloud security, illustrating how shared responsibility models, virtualization, and software-defined networking redefine both defensive and offensive strategies. These elements form the foundation for subsequent, more focused discussions on security features particular to AWS. This thesis will cover the architectural elements of AWS, including features like identity and access management, networking, and logging services. Through an exploration of how these elements integrate, it will unveil their importance in constructing successful threat detection systems.

A key element of this research entails examination of the attack techniques, tactics, and procedures (TTPs) employed by attackers on cloud infrastructures. Knowing TTPs allows for the design of patterns to detect anomalies that would otherwise pass undetected. It is not a case of simply copying known detection methods, but rather of tailoring them in forms that will be compatible with the dynamic and malleable nature of the cloud. To demonstrate this process, the thesis will identify specific attack vectors that target AWS, such as the misuse of access keys, privilege escalation, or lateral movement through misconfigured services.

The primary contribution of this thesis lies in the development and validation of detection

patterns for the chosen techniques. Drawing on mature frameworks like MITRE ATT&CK, this thesis aims to formally specify these patterns, explaining their scope, objectives, and real-world steps necessary for implementation. The tests will be conducted in controlled environments to ascertain the feasibility and accuracy of each proposed detection method. Lastly, this thesis aims to provide security practitioners with theoretical insights and also practical solutions, with the hope of having a better understanding of AWS-based threats and enhancing the security posture of organizations operating in public cloud environments.

I THEORETICAL PART

1 INTRODUCTION TO CLOUD COMPUTING

Cloud computing is the latest phase of information technology (IT) and has reshaped the way that enterprises acquire and utilize computation resources. It enables enterprise services to benefit from servers, storage devices, databases, networking capabilities, software applications, and analytics tools via internet-based services. The key characteristics of cloud computing are its on-demand capability, whereby users can access resources from any location at any time and incur little startup costs; scalability, which enables organizations to scale resources up or down to meet changing needs. It also provides a strong basis for the effective deployment and management of applications [1].

This framework offers a number of benefits. First, it provides cost-effectiveness by doing away with the necessity for large capital outlays in hardware; an organization simply pays for its use. Second, its scalability is another significant advantage that enables firms to adjust or modify their operations promptly to match market demand. Cloud computing enables innovation by providing access to cutting-edge technology and the capacity to rapidly build and deploy applications [1].

There are many problems associated with cloud computing. First, security is a big issue as organizations can face breaches and cyber-attacks as their data are kept outside the organization. Further, dependence on external service providers can create problems related to the reliability of services, control over data, and reduced control over key infrastructures [1].

Understanding cloud computing's complex issues, especially those that involve security, is crucial in setups like AWS. Weighing the balance between benefits and possible risks, organizations can devise measures to realize optimum potential of the cloud platforms while implementing strategies to protect their valuable digital assets, which may become necessary as utilization of the cloud services continues to grow [1].

1.1 Advantages of Cloud Computing

The implementation of cloud computing exhibits a lot of benefits that propel modern enterprises towards better efficiency and innovation [1]. Mentioned that leading advantages

are modernization and innovation, scalability, flexibility and reach, cost reduction, disaster recovery, data protection. Each of them is described in more details as follows:

1. **Modernization and Innovation:** Cloud computing is a driver of digital transformation as it enables organizations to improve their IT infrastructure and bypass the constraints of legacy environments. The cloud environment, with its exposure to new technologies such as artificial intelligence, machine learning, and big data analytics, offers a setting that is highly conducive to innovation. This, in effect, speeds up the creation and deployment of new applications and services at a pace never before, enabling businesses to remain competitive and respond swiftly to market changes [1].
2. **Scalability:** One of the greatest benefits of cloud computing is in the ability to adjust resources based on changing demand. In situations where an organization finds itself facing unexpected surges in user demand or, on the other end of the spectrum, a short-term reduction in workload, cloud offerings furnish the resources required for addressing the shifts. Such responsiveness allows organizations to maintain best-in-class operational performance without suffering the adverse effects of either overprovisioning or underutilization, with overall efficiency of operations being enhanced [1].
3. **Agility and Accessibility:** Cloud services are very flexible in that they enable users to access applications and data from nearly any place through network connectivity. Such extensive accessibility enables remote working and collaboration to be effortless, as teams can collaborate irrespective of their geographical locations. Furthermore, the cloud accommodates an extensive range of devices and platforms, thus enabling users to utilize the best available without fearing that they will encounter compatibility problems. [1].
4. **Cost Savings:** Cloud computing reduces capital spending significantly, as there is no longer a need to invest in huge hardware and infrastructure. Instead, an organization can opt for a pay-per-use model, where it pays only for what it uses. These CAPEX-to-OPEX shifts not only reduce overall spending but also make finances more predictable and flexible, thus providing businesses with better means to strategically invest their budget [1].

5. **Disaster Recovery:** In fact, the majority of cloud providers incorporate disaster recovery solutions that render the cloud a viable platform for recovering and storing business data. Disaster recovery solutions on the cloud provide permanent protection and storage of data in various strategic locations. This method significantly reduces the possibility of data loss due to natural disasters, hacking, or any other system malfunction, thereby ensuring organizational continuity and resilience by facilitating instant restoration of critical functions following a disaster. [1].
6. **Data Security:** Contrary to popular myths, cloud computing has the potential to strengthen data security via higher-level security protocols and adherence to standards by cloud service providers. The providers put significant financial investment in security technologies, including encryption, intrusion detection systems, and frequent security audits, to protect sensitive information. Additionally, the cloud platform enables centralized security policy management and access controls, providing uniform and strong security for all applications and data [1].

Last but not least, the benefits related to cloud computing transcend mere cost efficiencies and involve items such as modernization, scalability, flexibility, efficient disaster recovery mechanisms, and enhanced data protection. Such a benefit enables firms to support innovation, extend operations efficiently, and conduct business in a sound and secure manner within an increasingly changing digital and competitive environment.

1.2 Cloud Disadvantages

While cloud computing provides a wide variety of advantages, [1] defined nearly multiple disadvantages that must be remembered. Security and privacy threats, the requirement of a stable internet connection, vendor lock-in, and service outages are some of the main disadvantages. Remembering these is important if an organization is to make highly informed decisions and formulate effective mitigation strategies. The disadvantages are explained in more detail in following list

1. **Security and Privacy Risks:** The primary and most significant issue in cloud computing is the potential for security breaches and privacy invasion. When data is stored

externally, the organization must rely on cloud service providers to safeguard the data. Despite the implementation of sophisticated security protocols, there may be a few loopholes, which can expose the data to unauthorized access, cyber-attacks, or data leakage. Data protection law compliance in the cloud can also be harder, since organizations have to make sure that the cloud providers they utilize are compliant with all applicable legal and industry requirements to safeguard personal and propriety information [1].

2. **Internet Connection Requirement:** At its most fundamental level, cloud computing necessitates a constant and reliable internet connection. In the absence of a stable internet connection, customers are not able to fully exploit the advantages offered by cloud-based systems, which can lead to business processes being severely disrupted. This dependence can be particularly troublesome for regions with inferior or unreliable internet infrastructure, especially in instances of unscheduled service outage. Also, the operation of cloud-based applications depends necessarily on available Internet bandwidth and latency, which may impact user experience and efficiency of mission-critical business processes [1].
3. **Potential for Vendor Lock-In:** Vendor lock-in happens when an organization develops a high level of dependence on one cloud service provider, hence making it hard to change to other providers or transfer services to other platforms. Such dependence may inhibit operational flexibility and undermine bargaining power because changing to another provider may entail high costs, complicated data migration procedures, and potential compatibility problems. Also, the proprietary technologies and services offered by certain cloud providers may increase the potential for vendor lock-in, thereby reducing an organization's freedom to take advantage of best-of-breed solutions or benefit from competitive pricing and innovation offered by other vendors [1].

1.3 Types of Cloud Services

Platform as a Service (PaaS) is an evolution of the Infrastructure as a Service (IaaS) model, but with a greater level of abstraction such that end-to-end development and deployment processes can be hosted in the cloud. Higher than the infrastructure is application devel-

opment tools, middleware, database management systems, and runtime environments. This wide platform enables developers to concentrate on application development and deployment without concern for the management or upkeep of the underlying hardware or software elements. Platform as a Service (PaaS) in particular facilitates the automation of the development cycle, fosters collaboration among development teams, and facilitates practices such as continuous integration and deployment (CI/CD). It is most widely utilized in the creation of web applications, mobile applications, and microservices-based systems that demand high-velocity iteration and scalability [2].

1.3.1 Infrastructure as a Service

Infrastructure as a Service provides the basic building blocks required in cloud computing and provides virtualized computer resources over the internet. IaaS enables the provision of key infrastructure elements like virtual machines, storage drives, networking capabilities, and operating systems on a pay-as-you-use basis. This enables organizations to rent hardware resources without physical maintenance, thus facilitating scalability and flexibility in resource usage. But the users continue to have control over operating systems, applications, and storage; thus, IaaS will be more suitable for those organizations that require customized environments and greater control over IT infrastructure [2]. The common use cases would be website hosting, management of development and test environments, and storage and processing of bulk data. Another use case would be deploying own machine learning models as described in [3].

1.3.2 Platform as a Service

Platform as a Service is an extension of the Infrastructure as a Service (IaaS) model, and it brings a higher level of abstraction in order to provide holistic development and deployment within the cloud. PaaS includes, besides the baseline infrastructure, application development tools, middleware, databases, and runtime environments. Such a comprehensive platform enables developers to focus on the development and deployment of applications without worrying about management or maintenance of the underlying hardware or software components. PaaS especially helps speed up the development lifecycle, fosters collaboration among

development teams, and supports practices such as continuous integration and deployment (CI/CD). It is mainly used in the development of web applications, mobile applications, and microservices architectures, which require fast iteration and scaling [2].

1.3.3 Software as a Service

Software as a Service is the most abstract level of cloud service, where complete software applications are provided over the Internet. With SaaS, users do not need installation, management, or maintenance of software on their local devices, as applications are hosted and maintained by the service provider. Users can access SaaS applications via a web browser or even dedicated client applications, which assures them of easy access and reduces the complexity in managing the software. This can be well suited for end-users looking for a ready-to-use solution of different business functions, such as customer relationship management (CRM), enterprise resource planning (ERP), email services, and collaboration tools. SaaS has several advantages: automatic updates, scalability, and integration with other cloud services that enhance how productivity and operational efficiency are achieved. It's critical for organizations to understand the differences between IaaS, PaaS, and SaaS in order to make informed decisions about their cloud strategies [2].

Each of these service models offers different levels of control, flexibility, and management capabilities; this way, an organization can choose the alternative that best fits its technical specifications and operational objectives. In the context of this dissertation, it is important to understand these cloud service models in order to deal with security challenges and adversary identification on such open platforms as Amazon Web Services—AWS. Various service models introduce distinct security concerns and threat environments, thereby requiring customized security approaches to safeguard assets and maintain the integrity of operations conducted in the cloud. In other words, IaaS, PaaS, and SaaS are core service models that provide varying levels of infrastructure manageability and service deliverability in cloud computing. This can give organizations a better opportunity to optimize their cloud deployments for greater scalability, flexibility, and efficiency while addressing the specific security and operational challenges of each model [2].

2 AN OVERVIEW OF CLOUD SECURITY

Cloud computing has revolutionized the way organizations manage their IT infrastructure, providing unparalleled scalability, flexibility, and affordability. With an increasing number of organizations now moving critical workloads and sensitive data to the cloud, security assurance in such environments has become a concern of highest priority [4].

Cloud security represents a set of policies, controls, and technologies implemented for the protection of cloud-based systems, data, and applications. While the cloud brings both benefits and challenges intrinsic to its shared characteristics, advanced security features are proposed by cloud providers; the customer must, however, adapt his security practices in line with the unique architecture of the cloud [4]. Security in cloud environments is built on foundational knowledge described in [5].

In this chapter, we will be discussing some of the underlying principles regarding cloud security, the model of shared responsibility, looking into common threats in these environments, and strategies to reduce these vulnerabilities. The information is vendor-agnostic—meaning most concepts presented herein can be applied to virtually any cloud platform. Therefore, this high-level review should give readers an essential understanding of cloud security awareness, enabling them to understand what makes it so complex yet important [4].

2.1 The Shared Responsibility Model

One of the important principles of cloud security is the shared responsibility model. This model defines how cloud service providers and their customers share security responsibilities. The understanding of this model is an essential pre-requisite for a effective management of security in cloud environments [4].

The cloud service provider is, in general, responsible for the security of the underlying infrastructure, including physical hardware, networking, and virtualization layers. Customers, on the other hand, are responsible for securing their deployments on top of the cloud environment, including, among other things, their applications, data, user access, and network configurations. These responsibilities may be allocated differently depending on the cloud

service model in question: IaaS, PaaS, or SaaS [4].

For example, in an IaaS, the customer has complete control over the operating system, applications, and network settings; hence, the customer is responsible for their security. On the other hand, in the case of SaaS, the provider manages almost all the layers; therefore, the customer's primary care is on user access and protection of data [4].

This shared responsibility model emphasizes the fact that security is a partnership. Neglecting one side of the partnership could create vulnerabilities, so collaboration between the providers and customers is crucial [4].

2.2 Key Threats in Cloud Security

While cloud platforms provide significant advantages, they also present unique vulnerabilities. Below are some of the most critical threats faced in cloud environments:

2.2.1 Misconfigurations

The most frequent risks found in cloud computing environments involve misconfigurations. These may be poorly configured storage buckets, overmissive access policies, or unsecured application interfaces. These kinds of errors can expose sensitive information or allow unauthorized access [4].

Mitigation Strategies: One can reduce misconfiguration-related risk considerably by means of regular configuration assessment, using automated tools to detect vulnerabilities, and following proven security best practices [4].

2.2.2 Data Breaches

Data breaches continue to be a major threat in cloud computing. Whether due to insider threats, external attackers, or weak access controls, compromised data can result in substantial financial losses and damage to reputation. [4].

Mitigation Strategies: Encrypting data both at rest and in transit, implementing strong access controls, and conducting regular audits of data usage can help mitigate this risk [4].

2.2.3 Distributed Denial of Service (DDoS) Attacks

DDoS attacks attempt to flood a cloud service with too much traffic so that it becomes unreachable by legitimate users. They have the ability to disrupt business functions and result in huge financial loss [4].

Mitigation Strategies: Employing distributed architecture, utilizing content delivery networks, and deploying automated traffic filtering mechanisms can help absorb and mitigate such attacks [4].

2.2.4 Insider Threats

Insider threats involve malicious or accidental actions by employees or trusted individuals that compromise security. These threats are particularly challenging as insiders often have legitimate access to sensitive systems and data [4].

Mitigation Strategies: Monitoring user activity, enforcing least privilege access, and fostering a strong security culture within the organization are crucial to addressing insider threats [4].

2.2.5 Credential Theft

With users managing multiple accounts across platforms, the reuse of passwords and weak authentication mechanisms can lead to credential theft. Compromised credentials are a gateway for attackers to access cloud systems [4].

Mitigation Strategies: Enforcing multi-factor authentication (MFA), implementing identity federation, and using modern identity management platforms are effective ways to combat this issue [4].

2.3 Strengthening Cloud Security Posture

To maintain a strong security posture in cloud environments, organizations must adopt a combination of proactive and reactive measures [4].

2.3.1 Proactive Security Measures

Proactive measures aim to prevent threats before they materialize. Key actions include:

- **Access Control:** Implement multi-factor authentication and granular role-based access controls to limit unauthorized access.
- **Configuration Management:** Regularly review and update system configurations to ensure alignment with security best practices .
- **Data Protection:** Encrypt sensitive data and ensure robust backup policies to maintain data integrity and availability [4].

2.3.2 Reactive Security Measures

Reactive measures focus on identifying and responding to security incidents effectively. Key actions include:

1. **Incident Response Planning:** Develop comprehensive incident response plans tailored to cloud environments.
2. **Monitoring and Logging:** Utilize logging tools to monitor user activity and detect anomalies in real time.
3. **Threat Intelligence:** Leverage threat intelligence platforms to stay informed about emerging risks and vulnerabilities [4].

2.4 Key Principles of Cloud Security

Several principles underpin effective cloud security strategies:

- **Principle of Least Privilege:** Users and systems should have only the minimum level of access necessary to perform their tasks.
- **Defense in Depth:** Layered security controls reduce the likelihood of a single point of failure.
- **Continuous Improvement:** Regularly update security policies and practices to adapt to evolving threats.
- **Automation:** Automate repetitive security tasks to reduce human error and improve efficiency [4].

2.5 Conclusion

The landscape of cloud security continues to change, demanding that organizations reconcile agility with security. By comprehending the shared responsibility model, defining principal threats, and implementing comprehensive strategies, organizations can successfully secure their cloud environments. This chapter establishes a foundation for cloud security understanding, observing that the defense of cloud environments is more than surmounting technical challenges and is a strategic necessity. Later chapters will explore certain areas of cloud security in greater detail and will cover incident response and risk management [4].

3 AMAZON WEB SERVICES SECURITY CONTROLS

Amazon Web Services is one of the prominent cloud computing platforms that offer a wide variety of services to fulfill different business needs. Analysis will be done on its wide variety of service offerings in this section. This section provides detailed explanation of AWS services that focuses on cyber security of the environment. These services are divided into a number of categories depending on what specific area of security they serve. AWS contains dozens of services that have at least some connection to security; the most important services for the thesis are listed below.

3.1 Identity & Access Management

Services in this group focus on identity and access management in the AWS environment. Utilizing these services is crucial proactive step to avoid cyber security incidents.

3.1.1 AWS Identity and Access Management

AWS Identity and Access Management (IAM) is a fundamental security service that manages who and what resources users, groups, and roles can access in an AWS environment. By centralizing identity and access control, IAM ensures that each entity, both human and machine, possesses only the permissions necessary to perform its role. The method referred to as the principle of least privilege significantly minimizes the attack surface by avoiding both unauthorized behavior and unintentional misconfigurations that lead to exposing sensitive information [6].

IAM facilitates administrators in establishing permission boundaries, which define the upper limit of privileges assignable to any given individual or role. The permission boundaries have an important function in maintaining consistent and predictable configurations in various groups and endeavors. For instance, while someone might have the role of an administrator, defined permission boundaries might stop access to highly sensitive services or limit activity that might inadvertently compromise security vulnerabilities.

Alongside permission bounds, IAM facilitates the creation of groups, enabling administrators to manage the privileges of multiple users under one policy. The method strengthens access right assigning and access right editing; rather than granting privileges on an individual-by-individual basis, membership in a group effectively governs an authorized activity of a user in a more integrated and efficient way.

A concrete example of a restrictive policy designed to grant only the minimum necessary access is shown in Listing 3.1. This policy limits a user to listing and retrieving objects from a single Amazon S3 bucket, preventing any write or delete operations. Such a policy exemplifies how least privilege is realized in practice.

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": [  
7         "s3:ListBucket",  
8         "s3:GetObject",  
9       ],  
10      "Resource": [  
11        "arn:aws:s3:::example-bucket",  
12        "arn:aws:s3:::example-bucket/*",  
13      ]  
14    }  
15  ]  
16 }
```

Listing 3.1 Restrictive IAM Policy Example

Through systematic use of restrictive policies, permission boundaries, and groups, AWS IAM plays a pivotal role in safeguarding an organization's cloud environment, ensuring that resources remain protected from both inadvertent and malicious activities [6].

3.1.2 AWS IAM Identity Center

AWS IAM Identity Center, also known by its former title, AWS Single Sign-On, is an integrated system for identity and access management for multiple AWS accounts and multiple cloud applications. As an integrated portal for sign in, IAM Identity Center makes identity, access, and membership in groups considerably easier to manage. In addition to enabling ease of use, IAM Identity Center enhances security through encouraging consistent application of access policies [7].

A notable strength of AWS IAM Identity Center is in its ability to federate with external identity providers, such as Microsoft Active Directory, Okta, and any solution following the SAML 2.0 standard. Such federations allow organizations to take advantage of their current systems for identity management, thus lowering administrative overheads and minimizing configuration mistakes. In addition, the service supports Multi-Factor Authentication (MFA), giving an extra layer of protection against access by unknown individuals.

A concrete implementation example is the setup of a federation scenario where an enterprise identity provider authenticates users and then redirects them to AWS IAM Identity Center. The high-level authentication flow can be summarized in the following steps:

1. A user attempts to access an AWS application (for instance, the AWS Management Console).
2. The application redirects the user to the identity provider for authentication.
3. Upon successful authentication, the user is issued a security token by the identity provider.
4. This token is then exchanged with AWS IAM Identity Center, which verifies the user's identity and retrieves the associated permissions and role mappings.
5. The user gains access to the AWS application according to the defined roles and policies.

This protocol makes individuals have to verify only once through the corporate identity provider. Then, only necessary privileges are delegated by AWS IAM Identity Center according to the concept of least privilege. In combining administration of users and optimizing policy administration, AWS IAM Identity Center is vital in maintaining strong security in today's cloud systems [7]. A concrete example of SSO credential theft is described in the practical part of the thesis.

3.1.3 AWS Secrets Manager

AWS Secrets Manager is an enterprise-level and scalable solution designed for storage, maintenance, and retrieval of sensitive data, including but not limited to, database credentials, API keys, and various types of tokens. Its ability to provide access to sensitive data in an aggregated and specialized way through Secrets Manager avoids vulnerabilities due to accidental access and disclosure, something necessary for maintaining a safe operating environment. Secrets Manager also comes integrated with automated rotation capabilities for secrets, thus ensuring credentials are up to date and hence minimizing compromised-key-induced risk [8].

Three frequently used API calls illustrate the core functionality of AWS Secrets Manager:

1. **CreateSecret:** This call allows you to create a new secret, specifying its name and the sensitive data it contains. You can store credentials, encryption keys, or any confidential string. By centralizing the creation and encryption of secrets, this approach negates the need for manually distributing credentials across multiple systems [9].
2. **GetSecretValue:** Once a secret is stored, you can retrieve it using this API call. The secrets are always encrypted at rest using AWS Key Management Service (AWS KMS), and decryption occurs only when you request the secret's value. This safeguards the secret from potential data breaches, as it is never stored in plaintext on client-side systems [9].
3. **RotateSecret:** One of the most powerful features of AWS Secrets Manager is the ability to automatically rotate secrets. Regular rotation significantly reduces the window of opportunity for attackers, as it invalidates old credentials within a specified interval. The `RotateSecret` call can be configured to work with custom Lambda functions, enabling flexible rotation strategies across various types of secrets [9].

The incorporation of Secrets Manager in an organization's security system allows for safe storage, regular rotation, and direct access by authorized personnel or services. Such advanced and centralized approach to securing and managing secrets effectively reduces credential compromise threat and strengthens a strong and governed position in today's cloud environment.

3.2 Network Security

The services described in this section protect against types of attacks from the network.

3.2.1 AWS Web Application Firewall

The AWS Web Application Firewall (WAF) is an integral defensive component for web applications by carefully analyzing and moderating inbound web traffic on the application layer. Using AWS WAF, administrators have access to define both static (managed) and custom rulesets, enabling them to filter through recurring attack vectors such as SQL injection and Cross-Site Scripting (XSS) and to define custom discovery and response strategies according to unique threats. Such flexibility is found to be immensely valuable in scenarios where timely response is required and where frequent updates to recently evolved attack mechanisms have to be made [10].

A standout feature of AWS WAF is the ability to define custom detection rules [11]. These rules can identify malicious traffic that falls outside standard or commonly known attack patterns. For example, an organization concerned about a specific Command and Control (C2) URL could create a custom rule that inspects the URI path for suspicious patterns:

```
1 {
2   "Name": "MalwareC2Rule",
3   "Priority": 1,
4   "Statement": {
5     "ByteMatchStatement": {
6       "SearchString": "http://x.x.x.x/malwarec2.php",
7       "FieldToMatch": { "UriPath": {} },
8       "TextTransformations": [
9         { "Priority": 0, "Type": "NONE" }
10      ],
11       "PositionalConstraint": "CONTAINS"
12     }
13   },
14   "Action": { "Block": {} },
15   "VisibilityConfig": {
16     "SampledRequestsEnabled": true,
17     "CloudWatchMetricsEnabled": true,
18     "MetricName": "MalwareC2Rule"
19   }
20 }
```

Listing 3.2 Custom WAF Rule Example

When traffic matches this pattern, AWS WAF will block the request, helping to prevent

malicious C2 communications. This level of granularity is essential for incident response, as it allows teams to swiftly deploy new rules against emerging threats.

Alongside the provision of customized regulations, AWS WAF offers managed rule sets available through Amazon Web Services and several third-party vendors for greater protection. The rule sets include extensive collections of current vulnerabilities and undergo periodic updates to protect against recently discovered exploits. Administrators have access to include both managed rules and customized regulations to provide strong protection.

Regarding pricing, AWS WAF is on a pay-for-use system, where charges for every Web ACL, every rule deployed, and web requests processed are included. Such flexible pricing allows companies to adapt their defensive approach according to their needs, thus keeping wasteful expenditure to a minimum. Due to its built-in managed rule sets, bespoke detecting ability, and clear pricing, AWS WAF is always an integral part of any solution in the cloud.

3.2.2 Amazon VPC Security Groups

Amazon Virtual Private Cloud (VPC) Security Groups are virtual firewalls attached to instances in a VPC, controlling inbound and outbound network traffic in accordance with access configurations defined by the client. The flexibility in operation gives administrators authority to have strict controls in place, such that only necessary ports and protocol access is available to authorized network objects. Organizations, by using security groups, maintain the concept of least privilege, thus minimizing possible effects in case of any breach in their systems [12].

Security Groups operate on an individual-instance level, thus becoming an integral building block in a multiple-tier architecture. As default, they block all types of traffic, and administrators have to consciously define any authorized inbound and outbound transmissions. The approach avoids any unexpected vulnerabilities by configuration mistakes and makes them stronger in general. Moreover, their stateful behavior allows them to allow any return flow in response to prior accepted flows [12].

By adopting Amazon VPC Security Groups, businesses can maintain granular traffic gover-

nance, protecting their cloud-based assets in a cost-effective and scalable manner [12].

3.3 Security Monitoring & Threat Detection

Services described in this section focuses on security monitoring and detecting on suspicious events within the environment.

3.3.1 AWS Security Hub

AWS Security Hub is an integrated security management platform, enabling aggregation, categorization, and prioritization of data sourced through multiple AWS services and partner solutions in a single console. Leveraging data collected by various threat discovery tools, such as Amazon GuardDuty, Amazon Inspector, and Amazon Macie, Security Hub gives businesses an integrated view of their state of security. Consolidation of data through multiple threat discovery mechanisms makes possible the discovery of vulnerabilities, configurations, and possible dangers in the AWS platform, hence enabling threat assessment and consequent corrective measures [13].

A core strength of AWS Security Hub is in its seamless and holistic integration with a wide range of AWS services. For instance, if GuardDuty identifies an alert for suspect activity on an Amazon EC2, Security Hub collects and places such an alert in context in combination with alerts from several other security monitoring systems. The integrated view strengthens the correlation of events otherwise siloed in individual dashboards. The analysts therefore have better visibility to keep an eye on possible attack movement or configuration changes in various contexts, thus optimizing both detection and response [13].

Within the Security Hub, discovery is normalized and grouped by type, following the format reserved for AWS Security Findings, thus presenting consistent representation of security data. The format provides necessary details, such as severity, resource types, and timestamps, thus enabling teams to improve or accelerate their investigational workflows. Discovery also has the ability to forward data to services such as Amazon EventBridge, where such services may execute rule-based remediators, for instance, running an AWS Lambda function to isolate an affected instance [13].

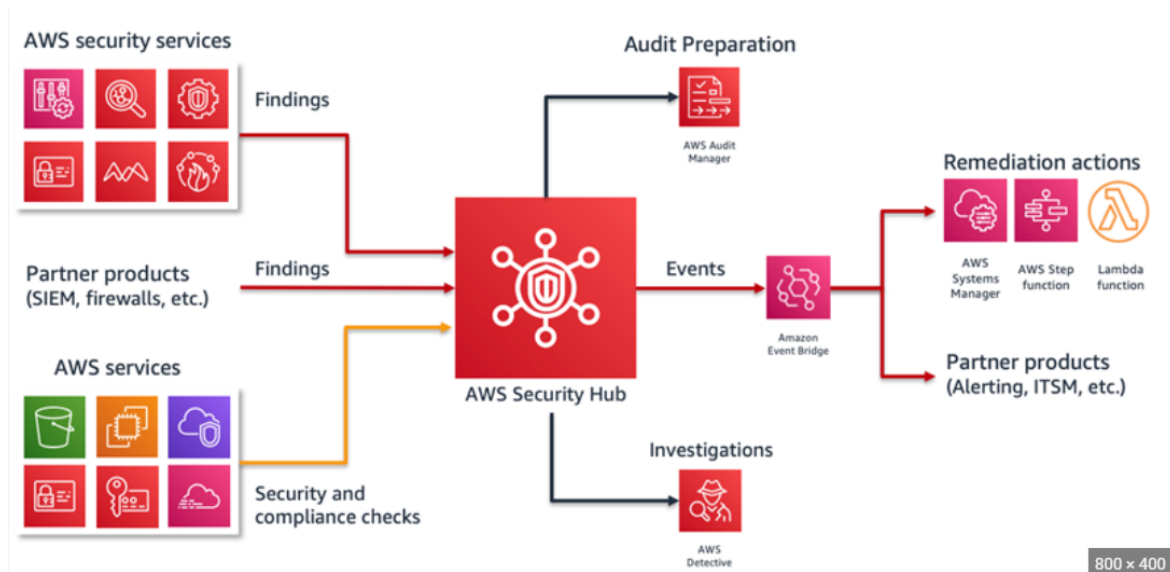


Figure 3.1 AWS Security Hub Integration With Other Services (from [14])

A noteworthy strength of AWS Security Hub is in its ability to produce and maintain context-dependent insights, including special queries or filters to prioritize given security findings. As an example, a team of security professionals might define a custom insight to identify root account use by staff who do not have legitimate reasons for using such accounts in multiple AWS contexts. An illustration of such an insight might include:

```

1 {
2   "Name": "RootAccountUsageInsights",
3   "Filters": {
4     "Title": [ { "Value": "root account", "Comparison": "
5       CONTAINS" } ],
6     "SeverityLabel": [ { "Value": "CRITICAL", "Comparison": "
7       EQUALS" } ]
8   }
9 }

```

Listing 3.3 Custom Insight Example

By surfacing critical root account usage findings, incident responders can promptly investigate and mitigate any high-risk actions performed by overly privileged identities.

Furthermore, Security Hub offers curated and managed insights. These insights are built, curated, and regularly kept up to date by AWS, including recognized best practices and recommended investigations. For instance, a managed insight might identify IAM users whose IAM policies are overly liberal and therefore deviate from documented best practices. If Security Hub identifies a policy giving unrestricted administrator access to an IAM user

who does not have multifactor authentication, it creates an equivalent insight. In an incident response workflow, such an insight might be used by security analysts to analyze access logs, network activity, and related findings in GuardDuty to identify if the IAM user conducted any malicious behavior. If such behavior is found, teams might choose to take back such a policy, enact multifactor authentication, or take other corrective actions to uphold the principle of least privilege [13].

Through the consolidation of security data, AWS Security Hub facilitates near-real-time correlations of findings and presents managed insights and context-optimized data to security professionals. The capability strengthens discovery, investigation, and threat response in today's cloud environments. In addition, by having strong interoperability for multiple AWS security services and third-party applications, it is an integral building block for building an enhanced and forward-thinking defensive posture for companies seeking to have an ongoing posture and reduce attack surfaces to their optimal limit [13].

3.3.2 Amazon GuardDuty

Amazon GuardDuty is an ongoing threat discovery system, constantly scanning AWS accounts, workloads, and data for signs of possible malicious activity. The system utilizes data aggregated from various sources, including Amazon VPC Flow Logs, AWS CloudTrail activity events, and DNS, to produce in-depth reports focusing on possible vulnerabilities. The ongoing monitoring approach lightens the load for security professionals by removing the need for them to constantly analyze large sets of log data, thus enabling them to focus on timely response and remediating [15].

GuardDuty can detect a wide range of suspicious activities, including anomalous API calls, unauthorized access attempts, and instances of data exfiltration. Nevertheless, it is important to note that GuardDuty is not designed to inspect the contents of payloads or perform in-depth application-level analysis. Instead, it focuses on behavioral indicators in network traffic and AWS account usage patterns to uncover potential threats [15].

Three examples of GuardDuty findings illustrate the breadth of its detection capabilities:

1. **UnauthorizedAccess:IAMUser/ConsoleLoginSuccess.B:** Triggered when a successful login occurs from an unusual geographic region or IP address, suggesting compromised credentials.
2. **Recon:EC2/PortProbeUnprotectedPort:** Indicates that an Amazon EC2 instance is under port-scanning attempts, signifying reconnaissance actions by potential adversaries.
3. **Trojan:EC2/DNSDataExfiltration:** Detects anomalous DNS requests suggestive of malware communication or data exfiltration through DNS tunnels.

The outcomes of such analysis make efficient prioritization of questions and implementation of corrective actions possible. The data collected by GuardDuty is also integrated in automated remediators using Amazon EventBridge and AWS Lambda. For instance, following discovery of a malicious sign-in, Lambda is able to initiate isolation of the compromised EC2 instance or de-activate compromised credentials. Such automated response compresses response time, thus preventing possible harm by ongoing attacks [15].

While GuardDuty offers advanced and pre-defined threat detection mechanisms, it is also shown to have flexibility in addressing unique needs of different organizations. Administrators have the choice to selectively include or exclude individual data sources, modify threat detection parameters between different AWS Regions, and define allowlists to reduce false alarms for recognized and sanctioned network IP addresses. Organizations can maintain an ideal balance between extensive monitoring and operational flexibility by fine-tuning such factors [15].

In summary, Amazon GuardDuty is an integral building block in an organization's approach to securing the cloud, enabling ongoing monitoring, automated alerts on threat activity, and effective response to incidents. Its ability to identify various types of malicious activity in AWS systems strengthens the overall defensive system, giving response teams sufficient time to address weaknesses and thwart larger breach attempts [15].

3.3.3 AWS CloudTrail

AWS CloudTrail is an extensive logging system that carefully captures in-depth details of every API request and following activity in an AWS environment. In capturing, maintaining, and enabling the encryption of any activity recorded, CloudTrail promotes greater organization accountability and transparency, hence becoming an integral piece in sound cloud security practices. The events logged include data on who performed an operation, on what resource, and on what time, and any supplementary data in accordance with operational and compliance needs [16].

A core strength of CloudTrail is its integrity in operation. Configured to maintain integrity in log files, every log file is given a cryptographic signature, and on regular intervals, a "digest" is computed. These "digest" files include checksums to allow integrity checks for changes in logged events after storage in Amazon S3. The integrity check system assures that, in case of any breach in security or in case of any audit for compliance, companies have trust in recorded events. Such cryptographic checks in regulated industries where tamper-evident logs are crucial for forensic analysis are frequently indispensable [16].

CloudTrail supports two primary categories of events: *management events* and *data events*. Management events encompass high-level operations such as creating or deleting an Amazon EC2 instance, modifying AWS Identity and Access Management (IAM) policies, or configuring security groups. These are actions that change the configuration of an AWS resource or that use control plane operations within the AWS Management Console or AWS CLI. A concrete example of a management event would be an IAM user creating a new policy with overly permissive permissions. Such a log entry would contain the user's ARN, the API call (CreatePolicy), and timestamps indicating when the call was initiated and completed. By reviewing this information, security teams can verify who made the policy change, whether it was authorized, and if further investigation is needed [16].

Data events, on the other hand, provide granular visibility into the operations performed on specific resources. One of the most common use cases for data events is to track object-level activities within Amazon S3 buckets. For example, data events would log the GetObject or PutObject operations when a file is accessed, uploaded, or modified. This level of detail

is especially valuable for detecting data exfiltration attempts or unauthorized file operations. By focusing on the actual data layer interactions, organizations can establish a fine-grained auditing mechanism that complements the broader oversight provided by management events [16].

In order to improve usability, CloudTrail offers the ability to collect and store logs in a centralized store known as CloudTrail Lake. This ability, in addition to enabling query-eligible activity data to be stored for extensive time intervals in multiple regions and accounts in AWS, gives investigators their own sandbox to query and analyze events in their history. Using CloudTrail Lake, threat investigators have the ability to query massive collections of events using SQL, thus enabling them to quickly identify malicious activity, correlate suspect activity, or execute retrospective threat hunting programs. [16].

A key strength of CloudTrail is its seamless interoperability with external SIEM systems. Many companies rely on SIEM systems to provide them with an integrated view of their entire IT system, including on-premises systems, third-party applications, and cloud services. CloudTrail events may be sent to Amazon S3, and then automated retrieval or streamed to SIEM systems in real time. The integrated solution allows for events in Amazon Web Services (AWS), network activity, endpoint protection system alerts, and telemetry data to be correlated by security analysts, thus giving them a single platform to identify abnormalities and respond to incidents. For example, if an on-premises network sensor for detecting intrusions identifies malicious network activity in concurrent time, while CloudTrail is reporting attempts to assume a given role, SIEM-provided correlation accelerates root-cause analysis and simplifies remediation [16].

In conclusion, AWS CloudTrail provides valuable insights on "who, what, what, and where" in interactions against multiple resources in AWS. Its ability to provide tamper-evident logs that cover data and management events, in addition to performing comprehensive analysis through CloudTrail Lake, makes reporting for compliance, security monitoring, and response efficient and convenient. The ability to incorporate CloudTrail data in external Security Information and Event Management systems strengthens the organization's threat detection and orchestration capabilities, thus solidifying CloudTrail's position as an integral piece of AWS security practices [16].

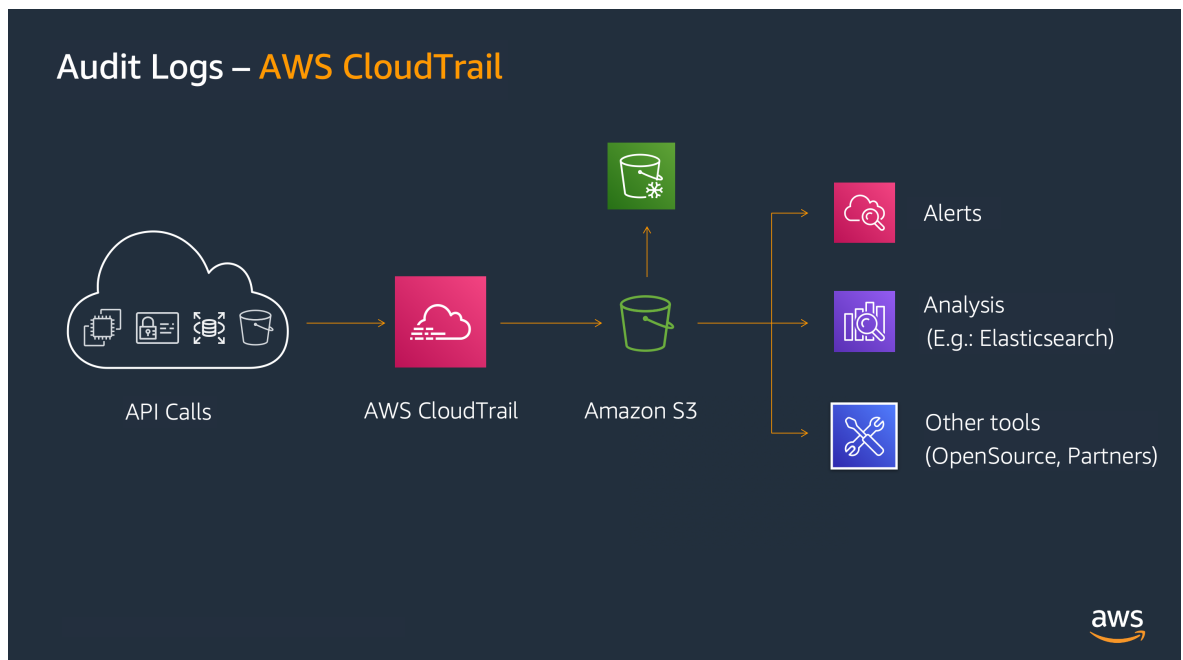


Figure 3.2 AWS CloudTrail (from [17])

4 EVALUATION OF AWS SECURITY SERVICES

This section focuses on the evaluation of AWS security services using a multi-criteria analysis based on the following weighted criteria:

- **Confidentiality** – 0.25
- **Integrity** – 0.20
- **Availability** – 0.20
- **Compliance** – 0.15
- **Cost** – 0.10
- **Scalability** – 0.10

Each AWS service is assigned a number of points for each security-related category. The ratings are scored on a scale from 1 (lowest) to 5 (highest). The services are grouped into various security-related categories, with each category including example services that fall into that group.

- **1 Point:** Minimal security features or capabilities. Services rated with a 1 offer only the most basic security controls and have very limited functionality.
- **2 Points:** Below-average security capabilities. Services with a 2 rating provide some security features, but they lack comprehensive protection and advanced functionalities.
- **3 Points:** Average security posture. Services rated with a 3 meet the baseline industry requirements by offering standard security features and protections.
- **4 Points:** Above-average security performance. A 4 rating indicates robust security features, strong protection measures, and effective integration with AWS security practices.

- **5 Points:** State-of-the-art security. Services with a 5 rating exhibit comprehensive protection, advanced security features, and best-in-class practices, setting the industry benchmark. Furthermore, complexity is manageable for most of the organization.

Identity and Access Management

Solutions for managing access control, authentication, and authorization. These services ensure that only authorized users and systems can access AWS resources.

- **AWS Identity and Access Management** – Core service for managing users, roles, and permissions.
- **AWS Single Sign-On** – Enables centralized access management across multiple AWS accounts and applications.
- **AWS Organizations** – Facilitates governance and policy enforcement across AWS accounts.

Incident Response

Tools and processes used to detect, analyze, and respond to security incidents in the cloud environment.

- **AWS CloudTrail** – Records account activity and API usage to support forensic analysis.
- **Amazon GuardDuty** – Threat detection service that continuously monitors for malicious activity.
- **AWS Security Hub** – Aggregates, organizes, and prioritizes findings from multiple AWS services and partners.

Compliance and Auditing

Services that assist with meeting regulatory requirements and maintaining adherence to internal security standards.

- **AWS Config** – Tracks resource configurations and evaluates compliance against desired states.
- **AWS Audit Manager** – Helps collect evidence for audits and assessments.
- **AWS Artifact** – Provides access to AWS compliance reports and agreements.

4.1 IAM Services

This section evaluates services inside the IAM category.

4.1.1 Confidentiality

Strengths:

- **Granular Access Control:** AWS (IAM) enables the creation of fine-grained policies that define who can access resources and how [6].
- **Federated Identity and Integration:** Amazon Cognito [18] allows secure user authentication through external identity providers (e.g., Google, Facebook), thereby enhancing authentication capabilities for mobile and web applications.

Weaknesses:

- **Risk of Misconfiguration:** Although the tools provide detailed control, incorrect policy settings or misconfigurations may compromise data confidentiality. Practical examples show that human error is a frequent source of vulnerabilities [19].

Assigned Score: 4/5

Justification: Overall, AWS offers robust mechanisms for data protection. However, the complexity of correct configuration and the risk of human error lower the potential for a perfect score.

4.1.2 Integrity

Strengths:

- **Auditing and Logging:** The AWS CloudTrail service integrates the logging of all IAM configuration changes, enabling detailed monitoring and analysis of events. This capability significantly supports the integrity of identities and access rights [16].
- **Verification Mechanisms:** Services such as AWS Directory Service [20] ensure identity integrity through synchronization with external directories and the use of verified protocols.

Assigned Score: 5/5

Justification: Owing to extensive auditing support and robust mechanisms for change management, this category achieves a high level of integrity. The primary risk lies in the operational complexity rather than in technical limitations.

4.1.3 Availability

Strengths:

- **Global AWS Infrastructure:** IAM and its associated services leverage AWS's redundant global infrastructure, which minimizes downtime and ensures service continuity [6].
- **Automatic Scaling:** The services are designed to support a high volume of API calls and user operations without performance degradation.

Weaknesses:

- **Dependency on Configuration:** Despite high inherent availability, misconfigured policies can result in access blocks or other operational issues, which may affect the overall system availability [19].

Assigned Score: 4/5

Justification: While the technical availability of AWS services is high, operational risks associated with policy configuration and management can occasionally result in accessibility issues.

4.1.4 Compliance

Strengths:

- **Certifications and Standards:** AWS IAM and associated services comply with numerous international security standards (e.g., ISO, SOC, PCI-DSS), which facilitates alignment with regulatory requirements [21].
- **Support for Role-Based Policies:** The ability to centrally manage and monitor policies helps organizations demonstrate adherence to both internal governance and external compliance obligations.

Weaknesses:

- **Audit Complexity:** Due to the complexity of configurations and the diversity of services, maintaining consistent compliance across the entire infrastructure can be challenging—especially for large organizations [19].

Assigned Score: 4/5

Justification: Although AWS provides a strong foundation for regulatory compliance, the need for regular audits and the complexity of service configurations slightly lower the overall score.

4.1.5 Cost

Strengths:

- **Free Core Service:** Core features of AWS IAM, including user and role management, policy-based access control, and MFA integration, are offered at no additional cost. This makes identity management accessible without significant financial investment.
- **Deployment Flexibility:** Most IAM capabilities are natively integrated into the AWS account structure, reducing the need for costly third-party identity providers or additional infrastructure.
- **Comparative Costs with Azure:** While AWS IAM's core is free, Microsoft Entra ID (formerly Azure Active Directory) offers a similar greater free tier for 50 000 MAU (monthly active user) [22] with \$0.0163 price for any additional user, making Azure potentially more expensive for advanced identity governance scenarios. However, Azure's tighter integration with Office 365 and Microsoft 365 ecosystems may reduce complexity and operational costs in Microsoft-heavy environments.

Weaknesses:

- **Operational Complexity:** More advanced implementations—such as the use of AWS Organizations, IAM Identity Center (AWS SSO), or Amazon Cognito—can increase overall costs due to the need for additional administrative overhead, monitoring, and staff training. For instance, Amazon Cognito incurs charges of \$0.015 per MAU after the first 10,000 users [18]. This makes Amazon Cognito cheaper for MAU, however, Entra ID offers five times higher limit for free tier.

Assigned Score: 4/5

Justification: Although the core IAM services are cost-effective, complex deployment scenarios and increased operational requirements can raise total implementation costs. When compared to Azure, AWS offers better cost efficiency at the basic level, but both platforms may incur significant costs depending on the scope of use and enterprise integration needs.

4.1.6 Scalability

Strengths:

- **Native Scalability Support:** AWS IAM and related services are designed for large-scale operations. Automation tools and APIs enable the management of hundreds to thousands of identities and access policies without significant performance degradation. Services such as AWS Organizations and IAM Identity Center [23] extend this access control into a multi-level access management structure across multiple accounts. Allowing to split the workload into smaller chunks and manage each piece for specific AWS Organization account as needed.
- **Integration with Other Services:** Seamless integration of IAM with other AWS security and operational services (such as AWS Organizations, CloudTrail, and Config) facilitates organizational growth by enabling scalable governance and control structures.

Weaknesses:

- **Increasing Complexity:** As the number of identities and policies grows, administrative complexity rises. This can lead to configuration errors and the need to invest in advanced automation tools and policy management frameworks to maintain control and compliance.

Assigned Score: 4/5

Justification: While the system is inherently designed for high scalability, the increasing complexity of identity and policy management introduces administrative challenges and may require additional investment in operational tooling.

4.1.7 Final Score

Final score assigned to the IAM is: $0.25 * 4 + 0.2 * 5 + 0.2 * 4 + 0.15 * 4 + 0.1 * 4 + 0.1 * 4 = 4.2$. Overall, AWS IAM performs very well. A slight downside for organizations is the increasing complexity of the environment, which may lead to significant misconfigurations and security issues, as observed previously [19].

4.2 Incident Response Services

This section evaluates the AWS services used for incident response, focusing on their ability to detect, investigate, and respond to cloud-native security threats. The evaluated services include AWS CloudTrail, Amazon GuardDuty, and AWS Security Hub. These services form the core of AWS's incident response capability, enabling evidence collection, threat intelligence, and centralized security visibility.

4.2.1 Confidentiality

Strengths:

- **Access Control and Encryption:** AWS CloudTrail logs are stored in Amazon S3 and can be encrypted using customer-managed KMS keys. Access is strictly governed through IAM and S3 bucket policies, allowing organizations to isolate sensitive log data [16].
- **Scoped Intelligence:** GuardDuty and Security Hub findings are context-aware and designed to reveal only relevant information per account and resource. Access to findings is restricted by identity-based and resource-based policies, supporting confidentiality at the operational level. [15], [13]

Weaknesses:

- **Limited Granularity of Findings Visibility:** GuardDuty does not allow organizations to customize which types of findings are visible to which accounts. In multi-account setups, all delegated administrators receive the same stream of findings, regardless of their relevance or severity. This limits the ability to restrict access based on the sensitivity of specific alerts or the role of each account.

Assigned Score: 4/5

Justification: Encryption and access control mechanisms are robust and well-integrated, ensuring that threat intelligence data remains protected. However, the current limitations in customizing visibility scopes across large organizations reduce flexibility in some high-security use cases.

4.2.2 Integrity

Strengths:

- **Immutable Logging:** CloudTrail logs can be configured with object lock to enforce write-once-read-many (WORM) protection, ensuring the immutability of audit trails for forensic purposes [16].
- **Automated Correlation and Validation:** Security Hub performs rule-based and machine learning-enhanced correlation of security findings, reducing false positives and ensuring consistent incident data integrity. [13]

Assigned Score: 5/5

Justification: The integrity of incident-related data is strongly preserved through cryptographic and storage-level controls, backed by consistent validation and correlation engines.

4.2.3 Availability

Strengths:

- **Service Redundancy:** All three services are deployed across multiple AWS Availability Zones and offer regional failover capabilities. CloudTrail supports multi-region trail creation, ensuring event collection continues even in regional outages. [16]
- **Always-On Monitoring:** GuardDuty and Security Hub operate continuously with no user-side infrastructure to maintain. This minimizes service disruption risks during an actual incident. [15], [13]

Assigned Score: 5/5

Justification: AWS ensures high operational availability for its incident response services through built-in redundancy and managed infrastructure. These services are designed for persistent uptime under all conditions.

4.2.4 Compliance

Strengths:

- **Pre-Certified for Security Standards:** All services are included in AWS's compliance scope for standards [21].
- **Integration with Compliance Tooling:** Security Hub findings can be forwarded to AWS Audit Manager or AWS Config to support compliance reporting and incident response validation. [13]

Assigned Score: 5/5

Justification: The services directly support regulatory compliance and auditing workflows and are part of AWS's broader set of compliant services, reducing barriers to certification.

4.2.5 Cost

Strengths:

- **Detailed Tiered Pricing:** GuardDuty applies tiered pricing for different data sources. In the *US East (N. Virginia)* region, the monthly rates are as follows [24]:
 - **CloudTrail Management Events:** \$4.00 per 1 million events
 - **VPC Flow Logs and DNS Logs:**
 - * First 500 GB/month: \$1.00 per GB
 - * Next 2,000 GB/month: \$0.50 per GB
 - * Next 7,500 GB/month: \$0.25 per GB
 - * Over 10,000 GB/month: \$0.15 per GB
 - **Malware Protection for EC2 and S3:** \$0.015 per GB of scanned EBS volume or S3 object
 - **EKS Audit Log Monitoring:** \$0.0020 per audit event
- **Free Tier Availability:** A 30-day free trial is available for new accounts for all protection plans, including malware and EKS coverage.

Weaknesses:

- **Cost Complexity and Scaling Risk:** Organizations with high-volume VPC traffic or frequent CloudTrail events can incur substantial monthly costs. For example, analyzing 40 million CloudTrail management events would cost \$160/month, and processing 3,000 GB of VPC/DNS logs could cost over \$1,600/month.
- **Comparison with Azure Defender for Cloud:** Microsoft Defender for Cloud offers a flat-rate pricing model. In its enhanced (Defender) plan, customers are charged:
 - \$15 per virtual machine/month
 - \$7 per SQL database/month

- \$4 per container registry/month [25]

For predictable workloads and consistent resource use, Azure's flat-rate model may offer more cost predictability, while AWS's usage-based model may be more efficient in low-volume or elastic environments.

Assigned Score: 3/5

Justification: GuardDuty provides granular and flexible pricing, but costs can escalate quickly in environments with high data ingestion or broad security coverage. Compared to Azure, the AWS model is advantageous for smaller or optimized workloads but requires constant cost monitoring in large-scale deployments.

4.2.6 Scalability

Strengths:

- **Designed for Multi-Account Environments:** Security Hub and GuardDuty natively support AWS Organizations, enabling centralized aggregation of findings from hundreds of accounts with minimal configuration effort.
- **Automatic Scaling:** These services automatically scale with data and account size, requiring no provisioning or performance tuning from the user.

Assigned Score: 5/5

Justification: AWS incident response services scale seamlessly across multiple accounts, workloads, and geographic regions, making them well-suited for both small teams and enterprise-scale deployments.

4.2.7 Final Score

The final score for Incident Response services is calculated as follows:

$$0.25 \cdot 4 + 0.2 \cdot 5 + 0.2 \cdot 5 + 0.15 \cdot 5 + 0.1 \cdot 3 + 0.1 \cdot 5 = 4.55$$

AWS provides a mature and resilient set of services for incident response. These services are architected to support rapid detection, forensic integrity, and centralized operational insight. While cost can become a limiting factor in large-scale deployments, the architectural benefits—particularly in scalability, availability, and compliance—justify a high rating.

4.3 Compliance Services

This section evaluates the AWS services focused on compliance and audit readiness, specifically AWS Config, AWS Audit Manager, and AWS Artifact. These services help organizations track resource configurations, gather and map evidence to control frameworks, and access compliance documentation.

4.3.1 Confidentiality

Strengths:

- **Access Controls and Encryption:** AWS Config records are stored in Amazon S3, where organizations can apply SSE-S3 or SSE-KMS encryption and restrict access through S3 bucket policies and IAM roles, keeping sensitive configuration data safe from unauthorized users [26].
- **Granular Evidence Permissions:** AWS Audit Manager lets administrators define who can view, upload, or modify collected evidence. This helps protect confidential information associated with compliance frameworks and audit documentation [27].

Weaknesses:

- **Integration with External Systems:** For data exported from AWS compliance tools into external or on-premises systems, additional measures (e.g., encryption and access control) may be required to maintain the same level of confidentiality.

Assigned Score: 4/5

Justification: Encryption and IAM-based controls offer solid confidentiality for compliance data within AWS. Additional diligence is needed when integrating with systems that do not inherit AWS's security mechanisms.

4.3.2 Integrity

Strengths:

- **Immutable Configuration Snapshots:** AWS Config stores historical versions of resource states, enabling forensic traceability of any configuration changes. This is essential for proving the integrity of critical resources during audits [26].
- **Automated Evidence Collection:** AWS Audit Manager systematically collects evidence for major industry standards (e.g., PCI-DSS, HIPAA, SOC 2). Once ingested, evidence cannot be altered without generating a corresponding audit trail, protecting data integrity [27].

Assigned Score: 5/5

Justification: AWS ensures that compliance data is versioned, traceable, and tamper-evident, providing strong integrity guarantees for audits and investigative work.

4.3.3 Availability

Strengths:

- **Redundant Storage and Failover:** Both Config and Audit Manager data typically reside in Amazon S3, benefitting from high durability and multi-Availability Zone redundancy. This ensures persistent access to compliance history, even in the face of regional outages [26].

- **Managed Service Architecture:** Since these are fully managed services, AWS handles capacity, patching, and failover, keeping downtime to a minimum and ensuring consistent availability [27].

Assigned Score: 5/5

Justification: Backed by AWS's robust infrastructure, compliance data remains accessible and resilient under diverse operational conditions.

4.3.4 Compliance

Strengths:

- **Built-In Frameworks:** AWS Audit Manager includes ready-made mappings for frameworks like GDPR, ISO 27001, SOC 2, and PCI-DSS, reducing the time and effort required to align AWS workloads with regulatory standards [27].
- **On-Demand Documentation:** AWS Artifact provides rapid access to compliance certifications and audit reports (e.g., SOC reports, ISO certificates), simplifying interactions with regulators and external auditors [28].

Weaknesses:

- **Customization for Specialized Requirements:** While AWS supplies numerous common frameworks, organizations with highly specialized or niche regulations may need to build and maintain custom control mappings, which can be time-consuming.

Assigned Score: 5/5

Justification: AWS offers strong compliance coverage for mainstream standards and seamless documentation access. Only unusual or highly complex standards may require added work.

4.3.5 Cost

Strengths:

- **Granular, Usage-Based Pricing for AWS Config:** AWS Config pricing depends on the number of Configuration Items (CIs) recorded and Config Rule evaluations performed. In the *US East (N. Virginia)* region:
 - \$0.003 per Configuration Item recorded
 - \$0.001 per Config Rule evaluation

For example, an environment with 10,000 resources and weekly evaluations of 5 Config Rules might cost around \$160/month (assuming 10,000 recorded CIs at \$0.003 each, plus 50,000 rule evaluations at \$0.001 each) [29].

Weaknesses:

- **Audit Manager Fees at Scale:** AWS Audit Manager charges a monthly fee based on the number of active frameworks and AWS accounts. Organizations subject to multiple regulatory standards or operating many AWS accounts can incur higher costs [30].
- **Comparison with Azure Policy and Microsoft Purview:** Azure Policy is offered at no additional charge for evaluating resource configurations in Azure [31]. However, it does not contain all functionalities AWS Config offers. Thus, it must be combined with other services with different pricing lists. Conversely, AWS's usage-based model can be more economical for smaller, highly optimized environments.

Assigned Score: 4/5

Justification: While AWS Config provides cost flexibility, charges for AWS Audit Manager can escalate in large enterprises under multiple frameworks. By contrast, Azure Policy's free base tier and potential bundle pricing for advanced features may be advantageous if an organization has significant Microsoft licensing investments.

4.3.6 Scalability

Strengths:

- **Organization-Wide Oversight:** AWS Config Organization aggregators can consolidate configuration data and compliance statuses across many AWS accounts, simplifying large-scale governance.
- **Automatic Scaling to Resource Growth:** As more AWS resources or frameworks are added, Config and Audit Manager scale accordingly without manual provisioning or performance tuning.

Assigned Score: 5/5

Justification: AWS compliance services handle expansions in resource count and multiple accounts seamlessly, making them well-suited for both small-scale operations and massive enterprise deployments.

4.3.7 Final Score

Using the same weighted formula (0.25 for Confidentiality, 0.2 for Integrity, 0.2 for Availability, 0.15 for Compliance, 0.1 for Cost, and 0.1 for Scalability):

$$\text{Final Score} = 0.25 \times 4 + 0.2 \times 5 + 0.2 \times 5 + 0.15 \times 5 + 0.1 \times 4 + 0.1 \times 5 = 4.50$$

AWS offers a powerful compliance services suite with good confidentiality, integrity, and availability characteristics. Pricing can be modest for small or efficiently optimized workloads, but the costs can escalate considerably in large or highly governed environments—especially when employing AWS Audit Manager with numerous frameworks. In the meantime, Azure Policy's basic level is free, and other compliance features can be packaged with other services and combined licencing, making it a compelling option in Microsoft-dominated environments.

CONCLUSION

The ratings of AWS IAM-based security services, incident response function, and compliance tooling display a broad and strong cloud security landscape. Incident response (4.55 overall rating) and compliance services (4.50 overall rating) both show high overall effectiveness, emphasizing AWS's strong architectural design and depth of feature. Particularly, the compliance category achieves top marks because it has inherent compatibility with leading regulatory frameworks, automatic collection of evidence, and simplicity of access to compliance reports—features that streamline audit processes and reduce organizational expenditures.

Nevertheless, these benefits have a minor drawback in that the multi-service nature of AWS and its fine-grained policy management can introduce complexity as companies scale. The increased complexity typically translates to extra charges for specialist staff, more comprehensive training programs, and a greater possibility of misconfigurations. From a cost perspective, the pay-as-you-use pricing model offered by AWS simplifies the management of small or highly optimized workloads but can escalate in larger, controlled environments—particularly where several accounts and frameworks are leveraged. On the other hand, Microsoft Azure tends to incorporate security and compliance capabilities within Microsoft 365 licensing, which may be more cost-effective for organizations deeply invested in the Microsoft ecosystem.

Despite the costs and complexities involved, the all-encompassing services offered by AWS, combined with their high performance in key security areas, make them an attractive choice for organizations seeking scalable, flexible, and compliance-based solutions.

5 DETECTION ENGINEERING CONCEPTS

Detection engineering is an advanced engineering discipline for threat-centric detection mechanisms designed to enable companies to properly defend against adversary attacks. Using thorough analytics of adversary behavior, systematic tuning mechanisms, and ongoing innovations in security tools, detection engineering makes it possible to identify advanced and hidden attacks in timely fashion. In addition, this discipline also demands an intimate understanding of on-premise and cloud infrastructures, and extensive insights into adversary TTPs. Therefore, detection engineering is an integral component of current cybersecurity practices, and by delivering data-driven insights for defensive strategies to execute, it empowers security professionals.

5.1 Security Information and Event Management

Security Information and Event Management (SIEM) systems provide centralized repositories for collection and correlation of data on an organization's posture for security. In particular, SIEM tools collect, standardize, and analyze data on log events and security events on diverse, disparate systems, such as firewalls, intrusion protection systems, endpoint protection systems, and cloud services, thus providing an integrated view of an organization's posture for security. The driving factor for SIEM adoption is usually the growing system complexity and their interconnectedness in today's cybersecurity scenarios. Rather than interrogating individual, disparate systems, security professionals rely on SIEM to collect and correlate data to provide an integrated view. Such integrated view makes data triage and investigational processes efficient, ultimately leading to a quicker response to emerging threats [32].

From a detection engineering standpoint, SIEM platforms provide a rich context to design and refine detection rules. By correlating events across diverse environments—including AWS and other cloud platforms—SIEM systems help security teams detect patterns of malicious behavior that would be difficult to identify in isolation. For instance, a suspicious login to a cloud resource can be cross-referenced with endpoint activity, network logs, or threat intelligence feeds, allowing analysts to confirm or refute potential compromises more accu-

rately. Consequently, effective utilization of SIEM technology enables more proactive and targeted detections, contributing significantly to robust incident response and threat hunting efforts [32].

5.2 Pyramid of Pain

The Pyramid of Pain is a theoretical structure representing how opponents encounter various classes of threat intelligence indicators and threat mechanisms. The pyramid divides such indicators into various tiers, in ascending order, beginning with low-level, malleable components on the bottom and topping out in advanced, behavior-oriented traits on top. The tiers, in ascending order, include: Hashes, IPs, Domain Names, Network/Host artifacts, Tools, and lastly, TTPs on top [32].

At the lower tiers, some of these indicators, for example, hash values for recognized examples of malicious code or unique IP addresses, are prone to manipulation by attackers. In using hash-based detection practices, attackers have no difficulty rewriting their code to produce a distinct hash. On the other hand, using an alternative domain name or IP is an inexpensive approach for most attackers. Thus, while such basic indicators have direct benefits for defenders, their longer-term strategic benefits are somewhat curtailed [32].

As individuals rise up through the tiered structure, Network/Host Artifacts or tools employed by attackers become progressively harder to bypass through discovery and identification. Finally, in the top tier, focus on TTPs leads attackers to execute sophisticated modifications in their strategies. For example, in an AWS system, discovery of an adversary's use of AWS IAM role switching, unusual activity using the AWS CLI, or unusual privileges on S3 buckets is an advanced type of discovery. Such behavior signatures progressively become harder to mask or forge, thus heightening attackers' struggles to adapt their strategies [32].

By establishing means to detect such elevated levels, in this case, TTPs, security professionals subject attackers to greater operational burdens, thus hindering their ability to evade discovery and realize their objectives [32].

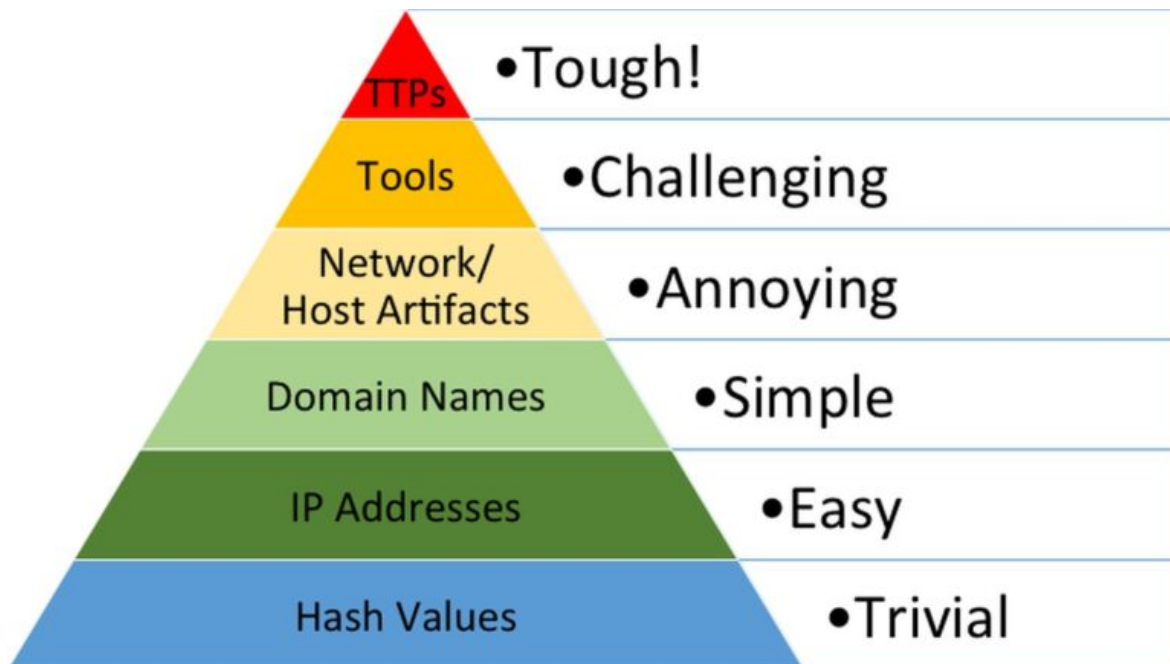


Figure 5.1 Pyramid of Pain (from [33])

5.3 Tactics, Techniques and Procedures

Tactics, Techniques, and Procedures define the “how” of adversarial operations. They are systematically described by frameworks such as MITRE ATT&CK, which categorizes adversary behaviors into a structured, hierarchical taxonomy. For cloud environments like AWS, MITRE provides a dedicated matrix of TTPs (see [34]) that helps security teams understand the techniques commonly employed by threat actors in cloud infrastructures. Nevertheless, attributing a specific tactic to a particular activity in a cloud setting can be challenging. Many actions—such as the manipulation of IAM roles—may simultaneously fulfill multiple objectives, blurring the lines between discrete tactics and making pinpoint attribution more complex [32]. An example list of concrete TTPs that attackers can perform in the AWS environment is below.

5.3.1 Valid Accounts

Threat actors routinely acquire and misuse legitimate account credentials to gain initial entry, maintain persistence, escalate privileges, or evade defenses. By exploiting these compromised credentials, adversaries can circumvent security controls and access critical resources across a network, including remote systems and externally accessible services such as vir-

tual private networks (VPNs), Outlook Web Access, network appliances, and remote desktop connections. These credentials may also grant elevated privileges to particular systems or allow entry into otherwise restricted sections of the network. Frequently, malicious actors opt to rely solely on valid credentials rather than deploy additional malware, thereby complicating attempts at detection [34]. In recent operations, the K3chang group used credential-stealing malware specifically designed to harvest valid login information, enhancing their ability to remain undetected and maintain unauthorized access [35]. An example scenario of using credentials stealers to obtain valid credentials is described in Section 7.5.

5.3.2 Create Account

Adversaries may establish new accounts to maintain their foothold within compromised systems [34]. By possessing sufficient privileges, they can create such accounts as secondary access points, eliminating the need to deploy additional persistent tools. Instances of such types may be found in localized systems, organizational structures, or in cloud systems. In general, attackers tend to exploit the AWS IAM system to access unknown accounts, thus lowering their chances of discovery.

The process of creation is usually an automated operation performed by authorized administrators in the cloud platform for granting access to recently included individuals; therefore, depending solely on creation events for identification may fail to capture necessary details. Therefore, creation events have to be correlated to other signs of malicious activity in order to identify possible malicious activity.

Below is an example of an AWS CloudTrail log demonstrating the creation of a new user (*persistenceuser*) within IAM. This type of API call can enable adversaries to retain unauthorized access to a targeted cloud environment:

```
1 {  
2   "eventVersion": "1.10",  
3   "userIdentity": {  
4     "type": "Root",  
5     "principalId": "000123456789",  
6     "arn": "arn:aws:iam::000123456789:root",  
7     "accountId": "000123456789",  
8     "accessKeyId": "ASIA5FTZFNQVQPXM4HQ4",  
9     "sessionContext": {  
10      "attributes": {  
11        "creationDate": "2024-10-17T08:05:34Z",
```

```

12     "mfaAuthenticated": "true"
13   }
14 }
15 },
16 "eventTime": "2024-10-17T08:08:23Z",
17 "eventSource": "iam.amazonaws.com",
18 "eventName": "CreateUser",
19 "awsRegion": "us-east-1",
20 "sourceIPAddress": "123.222.222.1",
21 "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
22             AppleWebKit/537.36 (KHTML, like Gecko)
23             Chrome/123.222.222.2 Safari/537.36",
24 "requestParameters": {
25   "userName": "persistenceuser"
26 },
27 "responseElements": {
28   "user": {
29     "path": "/",
30     "userName": "persistenceuser",
31     "userId": "AIDA5FTZFNQV6MQ6EXXFO",
32     "arn": "arn:aws:iam::000123456789:user/persistenceuser"
33   },
34   "createDate": "Oct 17, 2024 8:08:23 AM"
35 },
36 "requestID": "73187fc0-0a3b-4b87-9f83-7d1539117140",
37 "eventID": "e8ee5e43-8c4c-43c1-a66b-6fc730859acd",
38 "readOnly": false,
39 "eventType": "AwsApiCall",
40 "managementEvent": true,
41 "recipientAccountId": "000123456789",
42 "eventCategory": "Management",
43 "tlsDetails": {
44   "tlsVersion": "TLSv1.3",
45   "cipherSuite": "TLS_AES_128_GCM_SHA256",
46   "clientProvidedHostHeader": "iam.amazonaws.com"
47 },
48 "sessionCredentialFromConsole": "true"
49 }

```

Listing 5.1 Example of CreateUser event in AWS CloudTrail logs

5.3.3 Event Driven Execution

Adversaries possess capability to take advantage of built-in capabilities in AWS to maintain persistence or increase their privileges by running code in response to events. For instance, malicious actors might use Amazon EventBridge, otherwise referred to as CloudWatch Events, to invoke malicious AWS Lambda code. The malicious actors might maintain persistence or provide for movement between systems by simply creating or modifying an events rule pointing to compromised Lambda code, if only if running code holds elevated privileges.

Below is an illustrative AWS CloudTrail log event demonstrating how an adversary could alter EventBridge targets (PutTargets) to invoke a malicious Lambda function:

```

1 {
2   "eventVersion": "1.08",
3   "userIdentity": {
4     "type": "IAMUser",
5     "principalId": "AIDA5FTZFNQVUVMWRO7WR",
6     "arn": "arn:aws:iam::000123456789:user/testinguser",
7     "accountId": "000123456789",
8     "accessKeyId": "ASIA5FTZFNQVUKCGTHZD",
9     "userName": "testinguser",
10    "sessionContext": {
11      "sessionIssuer": {},
12      "webIdFederationData": {},
13      "attributes": {
14        "creationDate": "2024-10-23T16:26:23Z",
15        "mfaAuthenticated": "false"
16      }
17    }
18  },
19  "eventTime": "2024-10-23T16:33:38Z",
20  "eventSource": "events.amazonaws.com",
21  "eventName": "PutTargets",
22  "awsRegion": "us-east-1",
23  "sourceIPAddress": "123.222.222.1",
24  "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
25               AppleWebKit/537.36 (KHTML, like Gecko)
26               Chrome/123.222.222.2 Safari/537.36",
27  "requestParameters": {
28    "rule": "BackdoorEventRule",
29    "eventBusName": "default",
30    "targets": [
31      {
32        "id": "Idb7c7a53c-0595-47cc-8ce6-708fbb02e10c",
33        "arn": "arn:aws:lambda:us-east-1:000123456789:
34              function:BackdoorLambdaFunction"
35      }
36    ],
37    "responseElements": {
38      "failedEntryCount": 0,
39      "failedEntries": []
40    },
41    "requestID": "72062f38-6891-4d95-b194-40468ab0d773",
42    "eventID": "ed1f07d2-72d3-4f4f-8d82-8d034ae18f78",
43    "readOnly": false,
44    "eventType": "AwsApiCall",
45    "apiVersion": "2015-10-07",
46    "managementEvent": true,
47    "recipientAccountId": "000123456789",
48    "eventCategory": "Management",
49    "tlsDetails": {
50      "tlsVersion": "TLSv1.3",
51      "cipherSuite": "TLS_AES_128_GCM_SHA256",
52      "clientProvidedHostHeader": "events.us-east-1.amazonaws.
53    },
54    "sessionCredentialFromConsole": "true"
55  }

```

Listing 5.2 Example of PutTargets API call in AWS EventBridge logs

An example scenario illustrating detection possibilities for this TTP is presented in the practical section, specifically in Section 7.6.

5.3.4 Modify Authentication Process

Adversarial agents have the ability to compromise or manipulate standard authentication mechanisms, thus gaining prolonged or enhanced access to cloud systems while bypassing standard exploit strategies. In Amazon Web Services, such an approach highlights aspects concerning access and identity, including Identity Providers, trust models, and system authentication mechanisms. As such, through such systems, malicious actors have effectively bypassed protection mechanisms, lowered chances of discovery, and maintained unauthorized access to sensitive systems or data [34].

A prominent case in point is SAML provider metadata manipulation. SAML is an access federated protocol for authentication, enabling external Identity Providers to provide access to services in AWS through assertions. If an adversary is able to obtain sufficient privileges (i.e., by compromising an administrator's access to an AWS account), they have the ability to manipulate SAML metadata, such as injecting malicious attributes or modifying trust relationships. Such manipulation may allow attackers to forge SAML assertions or to impersonate authorized individuals' delegated roles, thus granting them access to sensitive data. Because such assertions appear to have valid authority, such an attack is sure to evade standard identity and access monitoring, thus enabling attackers to roam sideways and steal data in regions left unseen.

This TTP can fall under multiple categories in the MITRE ATT&CK framework, including *Credential Access*, *Defense Evasion*, and *Persistence*. By manipulating authentication data, adversaries effectively acquire credential-level privileges (Credential Access). At the same time, they circumvent security controls designed to detect illegitimate logins (Defense Evasion). Moreover, the ability to repeatedly assume unauthorized roles or reintroduce manipulated SAML metadata ensures a consistent foothold in the environment (Persistence). Consequently, organizations must apply a robust, multi-layered security strategy to prevent and detect such threats.

Detecting this TTP necessitates thorough visibility into authentication workflows and identity management changes. Organizations should implement strict version control and change management processes for SAML metadata and other trust relationship configurations. Mon-

itoring AWS CloudTrail logs for modifications to identity providers, unusual role assumption patterns, or suspicious policy updates can provide early warning indicators. Employing continuous compliance checks against known baselines (e.g., AWS Config or third-party security tools) also helps guard against unauthorized modifications. An example of this type of monitoring activity is described in Section 7.8.

Ultimately, defending against these attacks requires a combination of strong governance policies, least-privilege access principles, and robust monitoring capabilities that can detect anomalies in the authentication process. Failure to secure these mechanisms can result in persistent unauthorized access, privilege escalation, and the potential compromise of critical cloud resources.

5.3.5 Remote Services

Adversaries may exploit remote services in AWS to gain unauthorized entry, maintain persistence, and move laterally within a victim's cloud infrastructure [34]. This tactic frequently involves probing or abusing commonly used protocols such as Secure Shell (SSH) or Remote Desktop Protocol (RDP), which are configured to allow remote access. In Amazon Web Services, Elastic Compute Cloud (EC2) instances must have specific ports opened in *security groups* and *network access control lists* before an adversary can connect externally. For instance, enabling inbound traffic on TCP port 22 allows SSH connections, while port 3389 allows RDP.

Unintentionally, incorrect configurations in ACLs or in security groups may make EC2 instances available to the general population on the Internet, thus increasing vulnerability to brute-force or credential stuffing attacks. Furthermore, attackers who have compromised the environment might use such opened-up ports to execute lateral movements, thus expanding their access to other available resources. However, an intentional segregation approach through the implementation of Virtual Private Cloud (VPC) effectively limits available attack vectors for an adversary. Dividing up workloads and preventing direct external access, defenders can define possible attack vectors and limit any breach impact.

Regularly reviewing network configurations, enforcing least-privilege rules on security groups,

and closely monitoring suspicious inbound traffic are key measures to mitigate threats associated with exposed remote services in AWS.

An example activity related to this tactic is described in Section 7.10.

5.3.6 Credential Access

This technique involves obtaining credentials from centralized password management systems, thereby granting adversaries unauthorized access to sensitive resources [34]. In Amazon Web Services, attackers may exploit *AWS Secrets Manager*, a service designed to store and manage credentials securely. With elevated privileges, adversaries can list or retrieve various secrets (e.g., database credentials, API keys) and subsequently leverage these credentials to move laterally or escalate privileges within an environment [36].

A common attack scenario might begin with an adversary who has compromised a privileged AWS Identity and Access Management (IAM) user or role. Utilizing that access, they could invoke API calls such as `ListSecrets` to identify stored credentials, followed by `GetSecretValue` to obtain the actual password or key material. Further activities might involve `DescribeSecret` to gather additional metadata about the compromised secrets, potentially aiding in broader reconnaissance. Once the adversary possesses these sensitive values, they can use them to access databases, external services, or other protected assets, all while remaining undetected if the events go unmonitored [36].

Because these stolen credentials can directly facilitate unauthorized entry, this technique fits under the *Credential Access* tactic. Effective defenses include enforcing strict access controls on AWS Secrets Manager, regularly rotating stored secrets, and monitoring unusual API calls associated with secrets retrieval. By correlating suspicious `ListSecrets` and `GetSecretValue` events with unexpected user activity, security teams can identify potential credential compromise before it escalates into a more significant breach [36].

5.4 Sigma

Sigma is an open, community-driven project aimed at providing a standardized, vendor-neutral format for expressing detection rules in a structured and human-readable manner [37]. Originating from the need to describe security detections independent of a particular technology stack, Sigma rules enable security practitioners to define consistent detection logic across different SIEM solutions and analytics platforms. Instead of relying on platform-specific syntaxes, detection engineers can write rules that follow a unified template and then translate them to any supported SIEM query language [32].

The characteristics of portability and flexibility improve the effectiveness of detection engineering workflows. An organization is able to quickly adapt to novel tools or incorporate extra detection structures using a store of Sigma rules. In addition to interoperability, Sigma rules provide an organised and collaborative approach to detection logic. Further, versioning, testing, and validating of detection rules is made possible for security professionals, thus enabling reproducibility and traceability [32].

In the case of AWS implementations, the importance of Sigma is paramount. As systems and implementations transition to, or undergo, integration to cloud systems, detections' rationale needs to adapt to handle multiple data types and configurations of logging. The use of Sigma facilitates structured rationale in describing detections in context to cloud implementations, which in turn is compiled to query Amazon Security Lake, Amazon OpenSearch Service, or any external and cloud-native SIEM solution. Such flexibility enhances detection engineering, thus keeping organizations on their toes and in sync with the constantly changing threat landscape [32].

II PRACTICAL PART

6 ADVERSARY ACTIVITY DETECTION

This thesis discusses the application in practice of accurate detection rules using Sigma. The detection rules, encapsulating the logical structure of detection, may be specialized for presentation models to fit an organization's unique security solution. The detection rules in this thesis focus on the upper tiers of the Pyramid of Pain, focusing in on tools and TTPs. The lower tiers of the Pyramid of Pain, such as hash values, IP addresses, and domain names, fit better in threat intelligence feeds and may, for instance, be covered by Amazon GuardDuty.

Every environment is unique, and organizations have distinct procedures and handling methods. Consequently, detection rules must be tailored to the specific needs of each organization. The detection rules presented in this thesis primarily focus on TTPs, which inherently introduce a certain level of false-positive risk, as the targeted functionalities may be legitimately used within an organization. If a targeted functionality is essential for normal business operations, it is advisable either to exclude the rule altogether or to implement specific exceptions to filter out legitimate activity. These rules are built upon CloudTrail API telemetry and rely exclusively on API calls captured by this service.

6.1 Rule Structure

The structure of the detection rules is adapted from [32] and has been customized to align with the requirements of detection engineering in AWS environments. Each detection rule consists of the following fields:

1. **Title** - A human-readable title for the detection rule.
2. **Name** - A unique identifier for the detection rule, serving as a reference ID for correlation rules.
3. **Description** - A concise explanation of the rule's purpose, providing analysts with context when an alert is triggered.
4. **Author** - The name of the individual who created the rule, enabling organizations to identify the rule's origin and contact the author for inquiries or modifications.

5. **Creation date** - The date on which the rule was created.
6. **References** - Links to external resources that may provide useful context for investigating alerts.
7. **License** - The license under which the rule is distributed.
8. **Tags** - References to specific MITRE TTPs associated with the activity that the rule is designed to detect.
9. **Severity** - The priority level of alerts generated by the rule. For example, *low* severity typically indicates informational alerts requiring no immediate action; *medium* severity suggests potentially suspicious activity warranting further investigation; *high* severity signals a likely malicious threat requiring prompt action; and *critical* severity denotes an imminent and severe threat demanding an immediate and decisive response.
10. **False positives** - Known false positives associated with the rule.
11. **Investigation** - Recommended steps and guidelines for analyzing alerts generated by the rule to determine whether the activity is benign or malicious.
12. **Response** - Suggested response actions if the activity is confirmed to be malicious.
13. **Log source** - The log source utilized by the detection rule, which is essential when translating the Sigma rule into specific SIEM rules.
14. **Detection** - The core detection logic formatted according to Sigma standards.

Each detection rule references telemetry collected from CloudTrail and includes a corresponding translation to a specific SIEM solution to validate its effectiveness.

7 DETECTION RULES

This section presents a list of detection rules developed in this thesis. The rules follow the naming conventions outlined below:

1. **Keys** - YAML rule keys (excluding fields in the detection section) use kebab case.
2. **Detection names** - The names of detections use camel case.
3. **Detection parts** - Detection conditions within the detection section follow camel case notation.

Maintaining distinct notations for detection rule names and detection conditions within the rule itself allows for a clear distinction between references to other rules and specific conditions within a rule.

7.1 Shared Rules

To fully capitalize on the benefits of the detection-as-code approach, redundant code across multiple detection rules can be extracted into separate rules, which are then referenced within other rules in their condition sections. This section introduces shared rules that are referenced in later detection rules.

However, Sigma does not inherently support referencing other rules unless the primary rule type is set to *correlation*. To enable code reuse, a preprocessing script (see 0.31) was developed to modify the rules before they are processed by the Sigma engine. This script copies the detection section from the referenced rules and adjusts references accordingly, ensuring compatibility with the Sigma engine.

7.1.1 IAM Persistence

The rule shown in 7.1 defines a set of IAM service event names that may be used as part of persistence techniques. While these event names are not inherently malicious, their usage

under certain circumstances should be closely monitored. Specific cases where these events indicate potential threats will be covered by dedicated detection rules later in this thesis.

```

1 title: IAM Persistence
2 name: iam_persistence
3 description: "This detection identifies IAM events that can
  be used to establish persistence."
4 creation-date: 10/25/2024
5 author: Jan Blazek
6 license: MIT
7 tags:
8   - attack.TA0003
9 logsource:
10  product: aws
11  service: cloudtrail
12 detection:
13  selection:
14    eventName:
15      - CreateUser
16      - CreateAccessKey
17      - UpdateAccessKey
18      - CreateLoginProfile
19      - UploadSSHPublicKey
20      - CreateServiceSpecificCredential
21      - CreateRole
22      - UpdateAssumeRolePolicy
23      - AttachRolePolicy
24      - PutRolePolicy
25      - PassRole
26      - CreatePolicy
27      - AttachUserPolicy
28      - PutUserPolicy
29      - AttachGroupPolicy
30      - PutGroupPolicy
31      - CreatePolicyVersion
32      - CreateGroup
33      - AddUserToGroup
34 condition: selection

```

Listing 7.1 Sigma rule identifying IAM calls that may be used for persistence

7.1.2 Successful AWS Console Sign-in

This detection rule identifies successful sign-ins to the AWS console based on a combination of the event name, event source, and response elements.

```

1 title: Successful AWS Console Sign-in
2 name: successful_console_sign_in
3 description: "Detects successful sign-in to the AWS console"
4 author: Jan Blazek
5 creation-date: 10/29/2024
6 license: MIT
7 tags:
8   - attack.T1078
9 severity: Low
10 confidence: High
11 logsource:
12  product: aws
13  service: cloudtrail
14 detection:
15  selection:
16    eventName: ConsoleLogin
17    eventSource: signin.amazonaws.com
18    responseElements.ConsoleLogin: "Success"
19 condition: selection

```

Listing 7.2 Sigma rule identifying successful AWS console sign-in

7.1.3 Automated Execution

This detection rule identifies API calls executed via automation scripts. Automated scripts typically interact with AWS using an SDK (Software Development Kit), which by default includes custom user-agent strings. One of the most widely used AWS SDKs is Boto3 [38].

```
1 title: Automated API Call Execution
2 name: automated_execution
3 author: Jan Blazek
4 creation-date: 10/7/2024
5 confidence: High
6 detection:
7   selection:
8     userAgent|startswith:
9       - "Boto3"
10      - "aws-sdk-"
11 condition: selection
```

Listing 7.3 Sigma rule identifying API calls executed by automation scripts

7.2 Root Sign-in Without Multi-Factor Authentication

The AWS root user has unrestricted privileges within the environment, making it crucial to enforce multi-factor authentication (MFA) for this account to maintain the overall security of the AWS environment. AWS has established best practices for root user security [39], emphasizing the necessity of enabling MFA for the root user as a fundamental security measure.

Detecting root user sign-ins without MFA serves two primary purposes. First, it alerts the organization's security team that the AWS environment may not be compliant with established security standards. Second, it can function as an early warning mechanism in the event that an adversary has successfully disabled MFA and subsequently authenticated as the root user.

As a result, this detection rule plays a critical role in safeguarding the environment against potential compromises while ensuring adherence to security best practices.

```

1 title: AWS Root Sign in Without MFA
2 name: root_sign_in_without_mfa
3 description: "Detects when the AWS root user signs in without
  utilizing Multi-Factor Authentication (MFA), which poses
  a significant security risk as the root account has
  unrestricted access to all resources in the AWS
  environment."
4 author: Jan Blazek
5 creation-date: 10/29/2024
6 license: MIT
7 tags:
8   - attack.T1078
9 severity: Low
10 confidence: High
11 investigation: "Review the login details, including the
  source IP address, and location, to validate if the sign-
  in is consistent with known administrative patterns.
  Confirm whether the root account was expected to be used
  and check for concurrent suspicious activities.
  Additionally, audit other AWS services for any anomalous
  activity following the root login."
12 response: "Immediately enable MFA on the root account and
  rotate the root password. If unauthorized access is
  suspected, initiate a full account lockdown, including
  revoking active sessions. Rollback all changes made by
  root user following the successful login."
13 logsource:
14   product: aws
15   service: cloudtrail
16 detection:
17   selection:
18     userIdentity.type: Root
19     additionalEventData.MFAUsed: "No"
20   condition: selection and successful_console_sign_in

```

Listing 7.4 Detection Rule of Root Login Without MFA

The aforementioned rule can be translated into a SIEM-specific query to facilitate the detection of the targeted activity. Listing 7.5 presents the rule converted into a Splunk query using the Sigma framework. This query was generated by executing the command *sigma convert --without-pipeline -t splunk -f default preprocessed_rule.yml*.

It is important to note that the resulting query does not include the *index* field, as this field is specific to individual implementations. The *index* field can be configured using pipelines, which utilize the *logsource* attributes defined in the rule (refer to lines 13–15).

```

1 eventName="ConsoleLogin" eventSource="signin.amazonaws.com"
  userIdentity.type="Root" additionalEventData.MFAUsed="No"
  responseElements.ConsoleLogin="Success"

```

Listing 7.5 Root Login Without MFA Converted Into Splunk Query

7.3 Enumerate-IAM Tool

Enumerate-IAM is an open-source offensive tool available on GitHub [40]. The tool is designed to validate all event names and determine whether a user has the necessary permissions to execute them. This testing is conducted using a brute-force approach, with the order of executed API calls randomized, making correlation-based detection rules less effective.

The Enumerate-IAM tool is described by two detection rules. The first rule, shown in Listing 7.6, focuses on the payload *dummydata*, which is used as a placeholder value when testing API calls that require input parameters. The rule specifically targets two event names: *GetTransitGatewayMulticastDomainAssociations* and *GetVpnConnectionDeviceSampleConfiguration*, as these are unlikely to use *dummydata* as a legitimate input.

The same principle applies to any other event name originating from the *s3.amazonaws.com* event source. Since bucket names in Amazon S3 must be globally unique, legitimate API calls to *s3.amazonaws.com* using *dummydata* as a bucket name should, under normal circumstances, appear on only a single AWS account globally (or in a few other accounts that have shared access to that specific bucket).

```

1 title: Enumerate-iam Tool Signature (Unusual Payload)
2 name: enumerate_iam_bruteforce_multiple_calls
3 description: Detects unusual API call patterns commonly
  observed in the open-source project 'enumerate-iam', which
  is also included in penetration testing tools like Pacu.
  This rule covers multiple calls such as '
  GetTransitGatewayMulticastDomainAssociations', '
  GetVpnConnectionDeviceSampleConfiguration', and suspicious
  access to a bucket named 'dummydata'.
4 author: Jan Blazek
5 creation-date: 10/30/2024
6 references:
7   - https://github.com/andresriancho/enumerate-iam/tree/
  master
8   - https://github.com/RhinoSecurityLabs/pacu/blob/master/
  pacu/modules/iam__bruteforce_permissions/main.py
9 license: MIT
10 tags:
11   - attack.TA0043
12 severity: High
13 falsepositives:
14   Legitimate API calls targeting 'dummydata' for proof-of-
    concept or testing purposes, including '
    GetTransitGatewayMulticastDomainAssociations', '
    GetVpnConnectionDeviceSampleConfiguration', or bucket
    access.
15 investigation: >
16   Confirm whether the observed activity originated from an
    authorized user or application
17   by reviewing the IAM user's or role's assigned permissions
    and correlating the source IP
18   address with expected administrative activity. Evaluate the
    timestamps of the events to

```

```

19  identify patterns or unusual behaviors, and check for
    associated API calls in close time
20  proximity that could indicate a coordinated or scripted
    action
21  response: If unauthorized access is confirmed, revoke or
    restrict the permissions of the IAM role or user involved.
    Perform a review of other related cloud resources to
    identify further unauthorized modifications and secure the
    environment.
22  logsource:
23  product: aws
24  service: cloudtrail
25  detection:
26  selection1:
27  eventName: GetTransitGatewayMulticastDomainAssociations
28  eventSource: ec2.amazonaws.com
29  requestParameters:
    GetTransitGatewayMulticastDomainAssociationsRequest.
    TransitGatewayMulticastDomainId: dummydata
30  selection2:
31  eventName: GetVpnConnectionDeviceSampleConfiguration
32  eventSource: ec2.amazonaws.com
33  requestParameters:
    GetVpnConnectionDeviceSampleConfigurationRequest.
    VpnConnectionDeviceTypeId: dummydata
34  selection3:
35  eventSource: s3.amazonaws.com
36  requestParameters.bucketName: dummydata
37  condition: selection1 or selection2 or selection3

```

Listing 7.6 Enumerate-iam Tool Signature (Unusual Payload) detection rule

The second rule, shown in Listing 7.7, detects the use of the Enumerate-IAM tool based on an unordered sequence of unusual API calls. The following API calls were selected for detection due to their uncommon usage patterns:

1. **DescribeFpgaImageAttribute** - The use of FPGA (Field-Programmable Gate Array) resources is highly specialized and rare compared to general AWS services such as EC2, S3, or VPC. This API is relevant only for customers building hardware-accelerated workloads, representing a small subset of AWS users [41].
2. **GetInstanceTpmEkPub** - This API retrieves TPM (Trusted Platform Module) data from EC2 instances, a niche use case primarily relevant for secure computing. It is rarely encountered in most cloud environments [42].
3. **GetCoipPoolUsage** - COIP (Customer-Owned IP) pools are part of AWS Outposts and are only applicable to customers using hybrid solutions with custom IP ranges. As a result, its usage is restricted to a limited segment of AWS users [43].

If the tool is used without modification, both detection rules will trigger, leading to duplicate detections. However, covering the tool with multiple detection rules enhances the likeli-

hood of detection, even if the tool is modified to evade existing detection mechanisms. The detections referenced within the second detection rule are provided in Listing 7.7.

```

1 title: Enumerate-iam Tool Signature (Unusual API Calls
    Sequence)
2 description: "Detects an unusual sequence of API calls
    indicating permission brute force activity pattern
    commonly observed in the open-source project 'enumerate-
    iam', which is also included in penetration testing tools
    like Pacu."
3 references:
4   - https://github.com/andresriancho/enumerate-iam/tree/
    master
5   - https://github.com/RhinoSecurityLabs/pacu/blob/master/
    pacu/modules/iam_bruteforce_permissions/main.py
6 investigation: >
7   Confirm whether the observed activity originated from an
    authorized user or application
8   by reviewing the IAM user's or role's assigned permissions
    and correlating the source IP
9   address with expected administrative activity. Evaluate the
    timestamps of the events to
10  identify patterns or unusual behaviors, and check for
    associated API calls in close time
11  proximity that could indicate a coordinated or scripted
    action
12 response: If unauthorized access is confirmed, revoke or
    restrict the permissions of the IAM role or user involved.
    Perform a review of other related cloud resources to
    identify further unauthorized modifications and secure the
    environment.
13 author: Jan Blažek
14 creation-date: 10/30/2024
15 license: MIT
16 tags:
17   - attack.TA0043
18 severity: High
19 correlation:
20   type: temporal
21   rules:
22     - describe_fpga_image_attribute_event_name
23     - get_instance_tpm_ek_pub_event_name
24     - get_coip_pool_usage_event_name
25   group-by:
26     - userIdentity.arn
27   timespan: 15m

```

Listing 7.7 Enumerate-iam Tool Signature (Unusual API Calls Sequence) detection rule

7.3.1 Deployment

After preprocessing the rules using the Python script referenced in 0.31 and compiling them with the Sigma framework for deployment in a Splunk SIEM solution, the following detection rules are obtained:

```

1 | multisearch
2 | search eventName="DescribeFpgaImageAttribute" userAgent IN
    ("Boto3*", "aws-sdk-*") | eval event_type="
    describe_fpga_image_attribute_event_name" ]
3 | search eventName="GetInstanceTpmEkPub" userAgent IN ("Boto3
    *", "aws-sdk-*") | eval event_type="

```

```

    get_instance_tpm_ek_pub_event_name" ]
4 [ search eventName="GetCoipPoolUsage" userAgent IN ("Boto3*",
    "aws-sdk-*") | eval event_type="
    get_coip_pool_usage_event_name" ]
5
6 | bin _time span=15m
7 | stats dc(event_type) as event_type_count by _time
    userIdentity.arn
8
9 | search event_type_count >= 3

```

Listing 7.8 Enumerate-iam Tool Signature (Unusual API Calls Sequence) Converted Into Splunk Query

```

1 (eventName="GetTransitGatewayMulticastDomainAssociations"
    eventSource="ec2.amazonaws.com" requestParameters.
    GetTransitGatewayMulticastDomainAssociationsRequest.
    TransitGatewayMulticastDomainId="dummydata") OR (eventName
    ="GetVpnConnectionDeviceSampleConfiguration" eventSource="
    ec2.amazonaws.com" requestParameters.
    GetVpnConnectionDeviceSampleConfigurationRequest.
    VpnConnectionDeviceTypeId="dummydata") OR (eventSource="s3
    .amazonaws.com" requestParameters.bucketName="dummydata")

```

Listing 7.9 Enumerate-iam Tool Signature (Unusual Payload) Converted Into Splunk Query

7.4 Federation Token Abuse

Adversaries can exploit federated tokens to maintain access to an AWS environment even after their long-term access keys have been deactivated. This technique is described in [44]. When an attacker uses a federated token, revoking their access becomes significantly more challenging, as federated tokens are not easily deactivated or visible and can remain valid for up to 12 hours. AWS has published an official guide [45] detailing the process for revoking federated access, which should be followed during incident response activities.

According to [46], federated tokens are restricted from performing any IAM operations and all STS operations except for the *GetCallerIdentity* API call. However, this restriction does not apply when a federated token is used to sign into the AWS console, where IAM operations can still be executed. If an attacker attempts to exploit federated tokens to establish new long-term persistence after their original long-term access keys have been deactivated, the following sequence of four API calls must occur:

1. **GetFederationToken** - Used to exchange long-term access keys for a temporary federated token.
2. **GetSigninToken** - Generates a sign-in URL.
3. **ConsoleLogin** - Indicates a successful authentication using the previously generated sign-in URL.
4. **IAM service interaction** - An API call to establish new long-term persistence within IAM.

When analyzing the *ConsoleLogin* API call, it is important to note that the *userIdentity.type* field is set to *FederatedUser*. This combination of values implies that the *GetFederationToken* and *GetSigninToken* methods must have been invoked beforehand.

7.4.1 Detection Rules

Two detection rules were developed for this technique, designed to work together to identify and alert on suspicious activity. The first rule, *Federated User Performing IAM Persistence*, detects successful federated token abuse and raises an alert when any IAM operation indicative of potential long-term persistence is performed.

The second rule, *User Logging into the AWS Console Using a Federated Token*, detects an attack one step earlier—before IAM persistence is established. Additionally, this rule is designed to capture activity when an attacker uses an event not explicitly listed in 7.1.

Since both rules share the same logic for identifying federated users, this detection logic has been extracted into 7.10.

```
1 title: Federated User
2 name: federated_user
3 description: "This detection identifies operations performed
4             by a federated user."
5 author: Jan Blazek
6 creation-date: 10/25/2024
7 references:
8   - https://docs.aws.amazon.com/STS/latest/APIReference/
9     API_GetFederationToken.html
10 license: MIT
11 logsource:
12   product: aws
13   service: cloudtrail
14 detection:
```

```

13 selection:
14     userIdentity.type: FederatedUser
15 condition: selection

```

Listing 7.10 Sigma detection rule identifying operations performed by federated users

The final detection rules are presented in Listings 7.11 and 7.12. The confidence level of these rules is set to *medium*, as they may generate alerts for legitimate workflows under certain circumstances.

Tuning recommendations are provided in Section 7.4.2. Once the rule has been properly adjusted for a specific environment, its confidence level may be increased accordingly.

```

1 title: User Logging into the AWS Console Using Federated
  Token
2 description: "This detection identifies a user signing into
  the AWS Console using a sign-in URL generated by a
  federated token. Long-term security credentials can be
  used to request short-term federated tokens through the
  sts:GetFederationToken API. These federated tokens can
  continue to provide access even if the long-term security
  credentials that created them are deactivated. When used
  outside of the AWS Management Console, federated tokens
  cannot perform IAM operations unless explicitly permitted.
  However, these tokens can be used to generate a console
  sign-in URL, potentially bypassing such restrictions.
  Attackers have previously been using this technique to log
  into the AWS Console and bypass certain federation token
  restrictions related to creating long-term persistence."
3 author: Jan Blazek
4 creation-date: 10/25/2024
5 references:
6   - https://www.crowdstrike.com/en-us/blog/how-adversaries-
    persist-with-aws-user-federation/
7   - https://aws.amazon.com/blogs/security/how-to-revoke-
    federated-users-active-aws-sessions/
8 severity: Medium
9 license: MIT
10 falsepositives:
11   - "Legitimate operation when external authentication server
    is involved."
12 investigation: "Review if the activity is expected. Check
    other activity performed by the user after the successful
    login was made."
13 response: "Revoke federated users active session. If the
    usage of federated tokens is not needed if the environment
    , limit its usage using SCP."
14 tags:
15   - attack.T1078
16 logsource:
17   product: aws
18   service: cloudtrail
19 detection:
20   condition: federated_user and successful_console_sign_in

```

Listing 7.11 Sigma detection rule identifying federated user logging into the AWS console

```

1 title: Federated User Performing IAM Persistence
2 description: "This detection identifies IAM operations that

```

```

    may lead to long-term persistence being performed by a
    short-term token. Long-term security credentials can be
    used to request short-term federated tokens through the
    sts:GetFederationToken API. These federated tokens can
    continue to provide access even if the long-term security
    credentials that created them are deactivated. When used
    outside of the AWS Management Console, federated tokens
    cannot perform IAM operations unless explicitly permitted.
    However, these tokens can be used to generate a console
    sign-in URL, potentially bypassing such restrictions.
    Attackers have previously been using this technique to
    create new long-term access using short-term credentials
    even after the original long-term credentials were revoked
    "
3 author: Jan Blazek
4 creation-date: 10/25/2024
5 references:
6   - https://www.crowdstrike.com/en-us/blog/how-adversaries-
      persist-with-aws-user-federation/
7   - https://aws.amazon.com/blogs/security/how-to-revoke-
      federated-users-active-aws-sessions/
8 severity: High
9 license: MIT
10 falsepositives:
11   - "Legitimate operation when external authentication server
      is involved."
12 investigation: "Review if the activity is expected. Check
      other activity performed by the user after the successful
      login was made."
13 response: "Revoke federated users active session. Mitigate
      any persistence alerted by the rule. If the usage of
      federated tokens is not needed in the environment, limit
      its usage using SCP."
14 tags:
15   - attack.TA0003
16 logsource:
17   product: aws
18   service: cloudtrail
19 detection:
20   condition: federated_user and iam_persistence

```

Listing 7.12 Sigma detection rule identifying federated user performing IAM persistence

7.4.2 Excluding Legitimate Activity

The use of federated tokens also has legitimate use cases. The official AWS documentation for *GetFederationToken* [46] states that this API call is typically made by a server application acting as an external authentication system. If such a system is also designed to provide access to the AWS console, it may generate false positive alerts.

In this scenario, the recommended tuning procedure involves suppressing alerts based on *userIdentity.sessionContext.sessionIssuer.userName* for a predefined list of authorized users. Alternatively, if it is necessary to exclude all requests from a custom external authentication server, alerts can be filtered based on a combination of *sourceIPAddress* and *userAgent*.

7.5 Single Sign-On Credential Theft

AWS allows for the configuration of external Single Sign-On (SSO) authentication. This section focuses on detecting stolen SSO credentials when AWS CLI is used in conjunction with AWS IAM Identity Center [23].

7.5.1 SSO Authentication Workflow

Before developing detection mechanisms for this scenario, it is essential to outline the authentication workflow. The authentication process involves three primary API calls:

1. **CreateToken** - This API call generates access and refresh tokens for client authentication using client secrets [47]. The API supports three types of client secrets, each serving different purposes:
 - (a) **authorization_code** - Used when a client is redirected from one service to the SSO system for authentication, such as when executing the `aws sso login` command.
 - (b) **urn:ietf:params:oauth:grant-type:device_code** - Typically used on devices with limited user interaction. An example authentication flow can be found in [48].
 - (c) **refresh-token** - Used to obtain a new access token using a previously issued refresh token.
2. **GetRoleCredentials** - Once an access token is obtained via the *CreateToken* API call, it can be used to call *GetRoleCredentials*, which retrieves credentials for a role assigned to the IAM Identity Center user within a specific AWS organization [49].
3. **AssumeRoleWithSAML** - In the SSO authentication scenario described at the beginning of this section, *AssumeRoleWithSAML* is automatically invoked by AWS internal processes after a user successfully executes the *GetRoleCredentials* API call. This is indicated by the presence of the *AWS Internal* value in multiple log fields. More details can be found in [50].

7.5.2 AWS SSO and AWS CLI

This attack demonstration was conducted using the AWS Command Line Interface (CLI). The victim performed a standard AWS SSO authentication using the `aws sso login` command. Upon successful authentication, the AWS CLI stores multiple sensitive files within the `~/.aws/sso/cache` directory.

The structure of the stored JSON file is presented in Listing 7.13. This JSON structure contains secret information related to AWS SSO authentication. Additionally, credentials for specific roles are stored in the `~/.aws/cli/cache` directory.

```
1 {  
2     "startUrl": "https://d-9067c46113.awsapps.com/start",  
3     "region": "us-east-1",  
4     "accessToken": "accessToken",  
5     "expiresAt": "2025-02-06T09:11:08Z",  
6     "clientId": "clientId",  
7     "clientSecret": "clientSecret",  
8     "registrationExpiresAt": "2025-05-07T08:10:59Z",  
9     "refreshToken": "refreshToken"  
10 }
```

Listing 7.13 AWS CLI SSO Cache Data

```
1 {  
2     "ProviderType": "sso",  
3     "Credentials": {  
4         "AccessKeyId": "ASIA5FTZFNQV37DXY6SU",  
5         "SecretAccessKey": "secretAccessKey",  
6         "SessionToken": "sessionToken",  
7         "Expiration": "2025-02-02T19:37:38Z"  
8     }  
9 }
```

Listing 7.14 AWS CLI SSO Cache Data

7.5.3 Detection Strategies

A specific type of malware, known as an info stealer, is designed to extract secret information from a victim's computer. As described in [51], info stealers often target not only credentials stored within installed applications but also specific files on disk that may contain valuable information.

The goal of this detection strategy is to generate an alert when the entire `.aws` folder has been exfiltrated, allowing an attacker to gain access to a complete set of valid AWS credentials. Although there is no exact pattern that occurs when credentials are stolen, such an action is likely to result in anomalies within the logs, which could trigger detection rules.

The potential anomalies that may indicate credential theft are described below:

1. **Session token usage from different IPs** - An attacker leveraging stolen session tokens from the `~/.aws/cli/cache` folder will generate successful API calls originating from a different IP address. Since session tokens are unique per authentication session, this activity indicates that the same token is being used from multiple locations.
2. **Repeated refresh token usage** - If an attacker extracts a refresh token from the SSO cache, they can generate new access tokens. Monitoring for repeated usage of the same refresh token within 59 minutes (below the minimum access token expiration) may reveal an attack. However, AWS CloudTrail does not provide logs for this type of activity by default.
3. **Same access key usage from different locations** - Access keys are stored in both the CLI and SSO cache folders. Monitoring for instances where the same access key is used from different source IP addresses within a short period of time may help detect an attacker.
4. **Different user agent** - It is likely that an attacker will execute API calls using a different user agent than the original victim.

To effectively detect these anomalies, it is necessary to integrate them with the current detection capabilities provided by the Sigma format. Each of the described anomalies relies on detecting multiple distinct values for a specific field. Unfortunately, the Sigma format does not support specifying a field for threshold counting, making it impossible to create a detection rule that directly alerts on the anomalies described above.

The detection rule shown in Listing 7.15 alerts on the invocation of any event from AWS Identity Center service that occurs both from the AWS CLI and outside of the AWS CLI

within a 30-minute window. Additionally, log values for API calls provide the *userIdentity.credentialId* field, which is unique per access key. This detection mechanism is designed to identify scenarios where an adversary exfiltrates the SSO cache folder and subsequently uses the access key to invoke any event related to *sso.amazonaws.com*.

```

1 title: SSO Call From CLI
2 name: sso_call_from_cli
3 logsource:
4   product: aws
5   service: cloudtrail
6 detection:
7   selection:
8     eventSource: sso.amazonaws.com
9   cli:
10    userAgent|startswith: "aws-cli/"
11  --condition: selection and not cli
12  --
13 title: SSO Call Outside of CLI
14 name: sso_call_not_from_cli
15 logsource:
16   product: aws
17   service: cloudtrail
18 detection:
19   selection:
20     eventSource: sso.amazonaws.com
21   userAgent|startswith: "aws-cli/"
22  --condition: selection
23  --
24 title: Potential Access Key Theft
25 description: After a successful "aws sso login"
                authentication, AWS provides access and refresh tokens,
                which are stored on disk. This detection identifies an SSO
                access key from the AWS CLI being used outside of the AWS
                CLI, which may indicate that the access key has been
                stolen.
26 author: Jan Blazek
27 creation-date: 2/8/2025
28 severity: High
29 license: MIT
30 investigation:
31   - "Review source of the API requests and validate if the
        activity is legitimate."
32 response:
33   - "Sign out the compromised user from all session using '
        aws sso logout' command"
34   - "If needed, disable the user in the SSO service."
35 tags:
36   - attack.T1078.004
37 logsource:
38   product: aws
39   service: cloudtrail
40 correlation:
41   type: temporal
42   rules:
43     - sso_call_from_cli
44     - sso_call_not_from_cli
45   group-by:
46     - userIdentity.credentialId
47   timespan: 30m

```

Listing 7.15 Sigma detection rule identifying possible SSO access key abuse

Listing 7.16 presents the Splunk query obtained by converting the Sigma rule from Listing 7.15.

```

1 | multisearch
2 | search eventSource="sso.amazonaws.com" NOT userAgent="aws -

```

```
cli/*" | eval event_type="sso_call_from_cli" ]
3 | search eventSource="sso.amazonaws.com" userAgent="aws-cli/*
   " | eval event_type="sso_call_not_from_cli" ]
4
5 | bin _time span=59m
6 | stats dc(event_type) as event_type_count by _time
   userIdentity.credentialId
7
8 | search event_type_count >= 2
```

Listing 7.16 SSO Access Key Threft Converted Into Splunk Query Language

7.6 Lambda Executing Backdoor Code

One of the persistence techniques adversaries may use to maintain access to an AWS environment is the creation of a Lambda function designed to sustain their access. This technique has been incorporated into various offensive security tools and scripts, such as those referenced in [52] and [53].

Currently, there are three known types of backdoor Lambda code, each processing a specific event and performing a corresponding action. The trigger events are as follows:

1. **CreateUser** - The Lambda function generates access keys for newly created users and transmits the access key and secret key to a remote server.
2. **CreateRole** - When triggered, the Lambda function automatically updates the role's configuration to allow a specific user to assume the role.
3. **CreateSecurityGroup** - Upon triggering, the Lambda function modifies the security group configuration to allow traffic to or from a specified IP address.

7.6.1 Observed Anomalies

The use of backdoor Lambda functions results in minor log anomalies that may be detectable. Each backdoor Lambda function executes its action immediately upon receiving the trigger event. This behavior manifests in event logs, where an entity creation event is immediately followed by an update event originating from a different user or role modifying the same entity.

Another observable anomaly is the sequence of events leading to the creation of a backdoor Lambda function:

CreateFunction $\rightarrow$ *PutRule* $\rightarrow$ *PutTargets*.

This sequence of events signifies the creation of a new Lambda function, the establishment of an event rule to forward a specific event, and, finally, the configuration of the event rule to target the Lambda function.

7.6.2 Detection

Given the current detection capabilities of Sigma, it is not possible to create a strict detection rule for the first anomaly described, as Sigma does not support counting keys to alert on actions performed by different users within a short timeframe. The only feasible approach is to design a rule that does not validate distinct users. While this approach may generate false positives, it remains effective in detecting persistent Lambda functions already present in the environment.

The second pattern can be detected using a temporal correlation rule that identifies the expected *PutRule* pattern and associates it with *PutTargets*, linking the rule to a Lambda function. Listing 7.17 provides an example event pattern for detecting the *CreateRole* IAM function.

When designing this detection mechanism, it is important to consider alternative ways adversaries may implement the same functionality. For instance, attackers may specify only the *source* field without including *eventName* or *eventSource*. In such cases, all IAM events would be processed, with filtering occurring within the Lambda function itself.

```

1 {
2   "source": [
3     "aws.iam"
4   ],
5   "detail-type": [
6     "AWS API Call via CloudTrail"
7   ],
8   "detail": {
9     "eventSource": [
10      "iam.amazonaws.com"
11    ],
12    "eventName": [
13      "CreateRole"
14    ]
15  }

```

```

15     }
16 }

```

Listing 7.17 Event Pattern Processing CreateRole IAM Function

Another consideration is that *PutTargets* is plural, meaning that the function accepts an array of targets as a parameter. Listing 7.18 presents an example of request parameters used in a successful execution of the *PutTargets* function.

When deploying the detection rule in a production environment, it is essential to ensure that the backend correctly interprets the rule when the referenced field is encapsulated within an array.

```

1 {
2     "requestParameters": {
3         "rule": "BackdoorEventRule",
4         "eventBusName": "default",
5         "targets": [
6             {
7                 "id": "Idb7c9a53c-0585-47cc-8ce6-708fbd02e10c",
8                 "arn": "arn:aws:lambda:us-east-1:000123456789:function:BackdoorLambdaFunction"
9             }
10        ]
11    }
12 }

```

Listing 7.18 Example PutTargets Request Parameters

Listing 7.19 presents a detection rule that alerts on a sequence of events indicative of a suspicious event rule pattern associated with a Lambda function. The rule assumes that the *requestParameters.targets.arn* field matches any element within the targets array.

The suspicious event pattern is identified by processing all IAM events or at least one of the *CreateUser* or *CreateRole* event names. To minimize unnecessary false-positive alerts, the rule does not trigger if other IAM events are explicitly selected as part of the detection criteria.

A modified version of this rule, designed to detect an EC2-related backdoor, is provided in Listing 0.32.

```

1 title: IAM Event Rule Pattern
2 name: event_rule_pattern
3 logsource:
4   product: aws
5   service: cloudtrail
6 detection:
7   selection:
8     eventName: PutRule
9     eventSource: events.amazonaws.com
10    eventPattern|contains: "aws.iam"
11    event_name_key:
12    eventPattern|contains: "eventName"
13    event_names:
14    eventPattern|contains:
15      - "CreateUser"
16      - "CreateRole"
17    condition: (selection and not event_name_key) or (selection
18      and event_names)
19 ---
20 title: Event Targets Lambda
21 name: event_targets_lambda
22 logsource:
23   product: aws
24   service: cloudtrail
25 detection:
26   selection:
27     eventName: PutTargets
28     eventSource: events.amazonaws.com
29     requestParameters.targets.arn|contains: ":lambda:"
30   condition: selection
31 ---
32 title: Lambda Function Processing Suspicious IAM Events
33 description: "Lambda functions can be configured to trigger
34   execution when a specific event occurs in an AWS
35   environment. This detection identifies a Lambda function
36   processing suspicious IAM events. Threat actors have
37   previously been observed using Lambda functions to process
38   sensitive IAM events, potentially backdooring newly
39   created entities."
40 author: Jan Blazek
41 creation-date: 2/8/2025
42 severity: High
43 confidence: Low
44 license: MIT
45 investigation:
46   - "Review the the activity and verify if the activity is
47     legitimate."
48   - "Inspect the code inside the lambda function and validate
49     that the code is not malicious."
50   - "Investigate how the lambda function was created in the
51     environment."
52 response:
53   - "Revert all changes performed by the lambda function."
54   - "Perform containment functions on the lambda function and
55     its related entities."
56   - "Remove the event rule."
57 logsource:
58   product: aws
59   service: cloudtrail
60 correlation:
61   type: temporal
62   rules:
63     - event_rule_pattern
64     - event_targets_lambda
65 aliases:
66   rule_name:
67     event_rule_pattern: requestParameters.name
68     event_targets_lambda: requestParameters.rule
69 group-by:
70   - rule_name
71 timespan: 7d

```

Listing 7.19 Sigma Detection Rule for Lambda Backdoor

This Sigma rule can be converted only into a Splunk query, as Splunk is the only backend engine that supports both correlation rules and alias fields. The final query is presented in

Listing 7.20.

```

1 | multisearch
2 | search (eventName="PutRule" eventSource="events.amazonaws.
   com" eventPattern="*aws.iam*" NOT eventPattern="*eventName
   *") OR (eventName="PutRule" eventSource="events.amazonaws.
   com" eventPattern="*aws.iam*" eventPattern IN ("*
   CreateUser*", "*CreateRole*")) | eval event_type="
   event_rule_pattern" | rename requestParameters.name as
   rule_name ]
3 | search eventName="PutTargets" eventSource="events.amazonaws
   .com" requestParameters.targets.arn="*:lambda:*" | eval
   event_type="event_targets_lambda" | rename
   requestParameters.rule as rule_name ]
4 |
5 | bin _time span=7d
6 | stats dc(event_type) as event_type_count by _time rule_name
7 |
8 | search event_type_count >= 2

```

Listing 7.20 Splunk Query Identifying Lambda Backdoor

7.7 TruffleHog Abuse

TruffleHog is a security tool available on GitHub [54], designed to scan a company's source code for secret values embedded within the codebase. When used legitimately, this tool helps developers prevent accidental credential leaks in public repositories. However, there have been cases where TruffleHog was misused to scan third-party code in an attempt to harvest credentials that could provide unauthorized access to AWS infrastructure, as described in [55].

The objective of this detection mechanism is to alert on credentials discovered by the TruffleHog tool that do not originate from authorized activity. This means that if an attacker uses TruffleHog to scan the company's open-source repository, the company would receive an alert if the tool successfully identifies any valid credentials.

To develop this detection rule, it is necessary to determine how the tool can be uniquely identified. Upon inspecting the tool's publicly available source code on GitHub [54], it was discovered that the code snippet shown in Listing 7.21 is responsible for generating the user agent string used when the tool interacts with AWS.

Once TruffleHog detects a potential credential, it attempts to validate its authenticity by

invoking the *GetCallerIdentity* API function.

```

1 func userAgent() string {
2     if len(feature.UserAgentSuffix.Load()) > 0 {
3         return "TruffleHog " + feature.UserAgentSuffix.Load()
4     }
5     return "TruffleHog"
6 }

```

Listing 7.21 TruffleHog Code Creating User Agent

7.7.1 Detection

Since TruffleHog interacts with AWS services through a single API call to validate credentials, the only viable method for detecting its usage is by identifying the user agent pattern shown in Listing 7.21.

The detection rule in Listing 7.22 triggers an alert on any API call performed by TruffleHog. The invocation of an AWS API call signifies that the tool successfully identified secrets, which were subsequently used to make the request.

This detection rule leverages the *startswith* suffix to account for both cases presented in Listing 7.21.

```

1 title: Secrets Discovered by TruffleHog
2 description: "TruffleHog is a security tool that scans source
               code for potentially leaked credentials. This detection
               identifies a successful API call to an AWS environment
               from the tool, indicating that credentials may have been
               found. It can also help uncover potential unauthorized use
               of the tool."
3 author: Jan Blazek
4 creation-date: 2/10/2025
5 severity: Critical
6 license: MIT
7 references:
8   - https://github.com/trufflesecurity/trufflehog
9 investigation:
10  - "Validate whether the tool was executed by an
      unauthorized user."
11  - "Review other activities originating from the used
      credentials and investigate potential sources of the
      credential leak."
12 response:
13  - "Perform the necessary containment actions to prevent
      further use of the credentials."
14 tags:
15  - attack.T1078
16 logsource:
17   product: aws
18   service: cloudtrail
19 detection:
20   selection:
21     userAgent|startswith: "TruffleHog"
22   condition: selection

```

Listing 7.22 Sigma Detection Rule for TruffleHog

The Elastic Search Query Language (ESQL) representation of the TruffleHog detection rule is presented in Listing 7.23.

```
1 any where userAgent: "TruffleHog*"
```

Listing 7.23 TruffleHog Detection Rule Converted Into ESQL

7.8 Golden Security Assertion Markup Language Attack

The Golden Security Assertion Markup Language attack is a post-exploitation technique that allows an attacker to forge authentication assertions in a SAML authentication flow. First identified by [56], this method enables adversaries to generate valid authentication tokens without requiring a password or multi-factor authentication. By gaining access to a compromised identity provider private key, an attacker can sign malicious SAML assertions, granting themselves any role or privilege within a federated cloud environment.

This attack is particularly relevant to AWS environments that use SAML for Single Sign-On. In such cases, AWS relies on an external IdP to authenticate users and subsequently issues temporary security credentials based on the received SAML assertion. If an attacker compromises the IdP's signing certificate, they can craft arbitrary assertions to impersonate any user, including administrators, without triggering conventional authentication logs. This significantly undermines AWS security by enabling persistence, privilege escalation, and lateral movement within cloud infrastructures.

Within AWS environments, a Golden SAML attack may be executed using the *Update-SAMLProvider* API call by supplying malicious metadata. Without deeper insight into a specific company's infrastructure, it is not feasible to create detection rules that can reliably distinguish between benign and malicious SAML metadata.

Listing 7.24 presents a Sigma detection rule that alerts on a successful SAML provider update. The detection rule is assigned an *informational* severity level, as it may generate alerts for legitimate administrative activity. However, updating SAML provider metadata is considered a sensitive operation and should be actively monitored.

The detection mechanism focuses on the UpdateSAMLProvider API call targeting AWS IAM service. A successful execution is confirmed by checking for the presence of the response elements statement. If the operation fails, this field will not be present, and an *errorCode* field will appear in the log instead.

```

1 title: Successful SAML Provider Update
2 description: "Security Assertion Markup Language (SAML) is an
  open standard for authentication and single sign-on (SSO)
  . This detection identifies a successful SAML provider
  update. Threat actors have been observed modifying SAML
  provider metadata to take control of the authentication
  flow."
3 author: Jan Blazek
4 creation-date: 2/11/2025
5 severity: informational
6 license: MIT
7 investigation:
8   - "Review if the activity is authorized."
9   - "Validate metadata in the request and ensure they are
  legitimate."
10 response:
11   - "Restore the originate SAML metadata."
12   - "Reset credentials of all users that already
  authenticated via the malicious SSO."
13 tags:
14   - attack.T1556.007
15 logsource:
16   product: aws
17   service: cloudtrail
18 detection:
19   selection:
20     eventName: UpdateSAMLProvider
21     eventSource: iam.amazonaws.com
22     responseElements.SAMLProviderArn|contains: ":"
23   condition: selection

```

Listing 7.24 Successful SAML Provider Update Sigma Rule

Listing 7.25 presents the converted rule as a Splunk query.

```

1 eventName="UpdateSAMLProvider" eventSource="iam.amazonaws.com
  " responseElements.SAMLProviderArn="*:*"

```

Listing 7.25 Successful SAML Provider Update Rule Converted Into Splunk Query

7.9 Pacu

Pacu is an offensive security tool designed for testing AWS environments. The tool is publicly available on GitHub [53] and is maintained by Rhino Security. It consists of multiple modules that can be executed independently. This section introduces detection mechanisms for the Detection Disruption module, which is used to enumerate and disable security tools.

The Detection Disruption module in Pacu provides functionalities to manipulate AWS logging and monitoring services by disabling, deleting, or minimizing them. The following actions can be performed:

- **CloudTrail Trails**

- **Disable:** Stop logging for a specific trail.
- **Delete:** Completely remove the trail.
- **Minimize:** Modify settings to reduce logging without full deactivation. This includes:
 - * Removing associated SNS topics.
 - * Disabling global service event logging.
 - * Disabling multi-regional log collection.
 - * Disabling log file validation.
 - * Removing associated CloudWatch log groups and roles.
 - * Disabling log file encryption.

- **GuardDuty Detectors**

- **Disable:** Turn off the detector, stopping threat detection.
- **Delete:** Remove the detector from the account.

- **AWS Config**

- **Configuration Rules:** Allows deletion of specific rules.
- **Configuration Recorders:** Provides options to stop or delete recorders.

- **Configuration Aggregators:** Enables deletion of aggregators.
- **CloudWatch Alarms**
 - **Delete:** Remove specific alarms from monitoring.
- **VPC Flow Logs**
 - **Delete:** Remove flow logs associated with VPCs.

To perform these actions, Pacu first executes the ****Detection Enumeration Service**** module, which enumerates all detection services within an AWS environment.

The detection rule in Listing 7.26 alerts on the execution of the Detection Enumeration module when it attempts to enumerate all security services. The supporting rules that this detection relies on are provided in Listing 0.33.

```

1 title: Pacu Module Signature (detection__enum_services)
2 description: >
3   Detects a series of enumeration events initiated by the
4   Detection Enum Service module.
5   The module runs multiple "describe" commands against AWS
6   services (Config, CloudWatch,
7   EC2, and CloudTrail) to identify potential targets. The
8   detection correlates events
9   from the same user within a short timespan.
10  author: Jan Blazek
11  creation-date: 10/25/2024
12  references:
13    - https://github.com/RhinoSecurityLabs/pacu/blob/master/
14      pacu/modules/detection__enum_services/main.py
15  license: MIT
16  tags:
17    - attack.T1082
18  severity: Medium
19  falsepositives:
20    - "A cloud administrator enumerating AWS services for
21      threat detection purposes."
22  investigation: >
23    Verify whether the observed activity was executed by an
24    authorized user and review the
25    associated source IP address to determine its legitimacy.
26    Ensure that all monitoring alarms,
27    GuardDuty detectors, and CloudTrail trails are active and
28    functioning as expected.
29  response: >
30    Restrict the permissions of the compromised role or IAM
31    user to prevent further exposure.
32    In the event of a successful security disruption,
33    reconfigure all security services and review
34    the entire AWS infrastructure for unexpected changes that
35    the attacker may have performed while
36    logging was disabled.
37  logsource:
38    product: aws
39    service: cloudtrail
40  correlation:
41    type: temporal
42  rules:
43    - describe_trails

```

```

33     - list_detectors
34     - describe_alarms
35     - describe_flow_logs
36     - get_subscription_state
37     - config_enumeration
38 group-by: userIdentity.arn
39 timespan: 5m

```

Listing 7.26 Sigma Rule for Pacu Detection Enumeration Module

Most of the detection disruption activities are quite general and cannot be attributed directly to Pacu. The Sigma rule in Listing 7.27 alerts on minimization of a CloudTrail trail performed by the Detection Disruption module. The detection relies on specific combination of parameters passed to the *UpdateTrail* API function.

```

1 title: AWS Silence Trail
2 name: silence_trails
3 description: "Detects modifications to CloudTrail settings in
  a restrictive manner often employed by attackers to
  suppress defensive logging and alerting. The targeted
  trail may not record all critical events or forward them
  to other services as expected."
4 author: Jan Blazek
5 creation-date: 10/28/2024
6 references:
7   - https://github.com/RhinoSecurityLabs/pacu/blob/master/
  pacu/modules/detection__disruption/main.py
8 license: MIT
9 tags:
10  - attack.T1562
11 severity: Critical
12 confidence: Medium
13 falsepositives:
14   - "A cloud administrator updating the trail."
15 investigation: "Review whether the observed activity was
  performed by an authorized user by examining user
  permissions and correlating the source IP address with
  known administrative activity. Additionally, verify that
  all critical monitoring components such as CloudWatch
  alarms, GuardDuty detectors, and CloudTrail configurations
  remain active and correctly configured."
16 response: "Immediately restrict the permissions of the
  compromised IAM role or user to minimize potential damage.
  If a security breach is confirmed, restore and secure all
  relevant security services and conduct a comprehensive
  review of the entire AWS environment for any unexpected
  modifications made during the logging suppression period.
  Pay special attention to changes in global services like
  IAM and CloudFront."
17 logsource:
18   product: aws
19   service: cloudtrail
20 detection:
21   event:
22     eventName: UpdateTrail
23     eventSource: cloudtrail.amazonaws.com
24   params:
25     requestParameters.snsTopicName: ""
26     requestParameters.includeGlobalServiceEvents: false
27     requestParameters.isMultiRegionTrail: false
28     requestParameters.cloudWatchLogsLogGroupArn: false
29   condition: event and params

```

Listing 7.27 Sigma Rule for Pacu Trail Minimization

7.10 Unrestricted Access to Sensitive Ports

Threat actors can modify EC2 security groups to allow unrestricted access to sensitive ports, enabling them to maintain persistent access to AWS virtual machines. However, such modifications may not only be performed by attackers but can also result from misconfigurations. In both cases, the security team should be notified to assess the situation and restore security within the environment.

Allowing unrestricted access to sensitive ports violates AWS security best practices, as outlined in the AWS Well-Architected Framework [57]. To mitigate this risk, AWS Config provides a compliance check that flags security groups as non-compliant if certain ports are left open [58].

Listing 7.30 presents a Sigma detection rule that alerts on modifications to an EC2 security group that enable unrestricted SSH ingress traffic. Variants of this rule for RDP and SMB ports are provided in Listing 0.34. The corresponding Splunk queries for all converted rules are included in Listing 7.29.

The detection inspects the `AuthorizeSecurityGroupIngress` API call to the AWS EC2 service and verifies whether the desired port (22 in this example) falls within the port range specified by the `fromPort` and `toPort` request parameters. Unrestricted ingress traffic is checked by examining the `cidrIp` and `cidrIpv6` fields in the request; if either contains an unrestricted CIDR, the condition is met. A successful API call is indicated by the presence of valid response elements; if the call fails, the log will contain an `errorCode` instead of the response elements field.

```
1 title: "Unrestricted SSH Inbound Traffic"
2 description: "This detection identifies an EC2 security group
   allowing inbound SSH traffic from any IP address."
3 author: Jan Blazek
4 creation-date: 09/29/2024
5 license: MIT
6 tags:
7   - attack.T1562
8 severity: Medium
9 investigation:
10  - "Verify the reason for this activity with the user who
    performed the operation."
11  - "Review SSH logs from the EC2 instance associated with
    the affected security group."
12 response: "Restore a secure security group configuration."
13 logsource:
14   product: aws
```

```

15 service: cloudtrail
16 detection:
17   selection:
18     eventName: AuthorizeSecurityGroupIngress
19     eventSource: ec2.amazonaws.com
20     requestParameters.fromPort|lte: 22
21     requestParameters.toPort|gte: 22
22     responseElements.requestId|contains: "-"
23   ipv4Unrestricted:
24     requestParameters.cidrIp|contains: 0.0.0.0
25   ipv6Unrestricted:
26     requestParameters.cidrIpv6: ::/0
27   condition: selection and (ipv6Unrestricted or
    ipv4Unrestricted)

```

Listing 7.28 Sigma Detection Rule for Allowing Unrestricted SSH Ingress Traffic

```

1 eventName="AuthorizeSecurityGroupIngress" eventSource="ec2.
  amazonaws.com" requestParameters.fromPort<=22
  requestParameters.toPort>=22 responseElements.requestId="
  *-*" requestParameters.cidrIpv6="::/0" OR
  requestParameters.cidrIp="*0.0.0.0*"
2
3 eventName="AuthorizeSecurityGroupIngress" eventSource="ec2.
  amazonaws.com" requestParameters.fromPort<=3389
  requestParameters.toPort>=3389 responseElements.requestId="
  *-*" requestParameters.cidrIpv6="::/0" OR
  requestParameters.cidrIp="*0.0.0.0*"
4
5 eventName="AuthorizeSecurityGroupIngress" eventSource="ec2.
  amazonaws.com" requestParameters.fromPort<=445
  requestParameters.toPort>=445 responseElements.requestId="
  *-*" requestParameters.cidrIpv6="::/0" OR
  requestParameters.cidrIp="*0.0.0.0*"

```

Listing 7.29 Unrestricted Ingress Access to Sensitive Ports Splunk Query

7.11 High Volume of Unusual API Calls

This section focuses on building a detection rule for scenarios where a high volume of unusual API calls is used. As an example, this section will use execution of *GetPasswordData* from Status [59]. The Stratus simulation executes the API call on multiple different EC2 instances, simulating an attacker stealing RDP credentials from multiple Windows hosts.

Detection Sigma rules for these types of scenarios is to use *event_count* correlation type, as depicted in Listing 7.30. It is also necessary to set a correct threshold and timespan, depending on the environment.

This type of detection may also be covered by anomaly detections using machine learning as described in [60].

```

1 name: GetPasswordData API Call
2 name: get_password_data_api
3 detection:
4   selection:
5     eventName: GetPasswordData
6     eventSource: ec2.amazonaws.com
7   condition: selection
8 ---
9 name: High Volume of GetPasswordData Call
10 description: "This detection identifies a high volume of
    GetPasswordData calls originating from the same user
    within a short time interval. Threat actors may use this
    technique to retrieve RDP passwords for a large number of
    Windows EC2 instances."
11 author: Jan Blazek
12 creation-date: 02/15/2025
13 license: MIT
14 tags:
15   - attack.TA0006
16 severity: Low
17 falsepositives:
18   - "A cloud administrator retrieving passwords from multiple
    instances"
19 investigation: "Verify if the activity is authorized."
20 response:
21   - "Reset credentials of the targeted machines."
22   - "Perform necessary containment actions."
23 logsource:
24   product: aws
25   service: cloudtrail
26 correlation:
27   type: event_count
28   rules:
29     - get_password_data_api
30   group-by:
31     userIdentity.arn
32   timespan: 10m
33   condition:
34     gte: 15

```

Listing 7.30 Sigma Detection Rule for Allowing Unrestricted SSH Ingress Traffic

7.11.1 Advantages of Machine Learning for Anomaly Detection

Machine learning-based anomaly detection offers several advantages over static rule-based detection mechanisms:

- **Adaptability:** Machine learning models can adapt to evolving attack techniques and behavioral patterns without requiring manual updates to rules.
- **Reduced False Positives:** Traditional static rules may generate high false positives due to hardcoded thresholds, whereas ML-based models can dynamically learn baseline behaviors and identify truly anomalous activities.
- **Automated Threshold Tuning:** Unlike manually set static thresholds, ML models can dynamically adjust sensitivity levels based on the environment and historical data.

These advantages make machine learning a valuable complement to static rule-based approaches, enhancing the effectiveness of anomaly detection in cloud environments.

CONCLUSION

The multi-criteria analysis of selected AWS services—Identity and Access Management, Incident Response, and Compliance—has demonstrated the considerable strengths of AWS in delivering robust and adaptable security solutions. The results, which assign high ratings to both incident response (4.55 overall) and compliance capabilities (4.50 overall), reflect AWS’s consistent architectural focus on defense-in-depth and feature richness. These qualities simplify the adoption of recognized regulatory frameworks, expedite the process of gathering audit evidence, and offer seamless access to compliance reports. Consequently, organizations can reduce operational overhead by minimizing time-intensive manual tasks and focusing on higher-level security strategies.

Nevertheless, these advantages are partly offset by the complexity introduced through AWS’s fine-grained service model. Enterprises that expand their cloud footprint must navigate a potentially steep learning curve, often requiring specialized personnel and elaborate training programs. Moreover, cost factors can become non-trivial, particularly when managing high-volume workloads or multiple AWS accounts in large-scale environments. Although AWS’s pay-as-you-go model can be beneficial for smaller or optimized use cases, Microsoft Azure occasionally presents bundled security and compliance services within existing Microsoft 365 licensing, proving to be more cost-effective for organizations deeply invested in the Microsoft ecosystem. Despite such considerations, AWS remains compelling for entities seeking scalable, flexible, and compliance-focused infrastructure offerings.

From an operational perspective, the inherently dynamic nature of cloud ecosystems was further exemplified through the analysis of attacker TTPs. Assigning a single action to a single TTP proved to be highly challenging, given that multiple cloud-based events often map to more than one technique or phase of the attack lifecycle. This multi-dimensional relationship underscores the need for adaptable detection strategies, as rigid rule sets can fail to capture the subtleties of sophisticated threat actors operating in elastic environments such as AWS.

During the creation of Sigma rules for threat detection, it became evident that traditional,

non-correlational approaches to rule definitions pose significant limitations. Specifically, these approaches do not allow for direct references to other rules, thereby restricting the capacity to reuse or layer detection logic. To address this constraint, a custom Python script was developed to preprocess Sigma definitions. Through this mechanism, logically related detection patterns could be unified at deployment time, ensuring a consistent framework for identifying complex attack chains or multi-step exploits. This preprocessing stage reflects a crucial methodological improvement, enabling security teams to maintain a single, coherent repository of detection logic that remains easier to manage and update over time.

The thesis also identified and rigorously tested eleven specific attack techniques and tools—among them, *Root Sign-In Without MFA*, *Enumerate-IAM Tool*, *Federation Token Abuse*, and *High Volume of Unusual API Calls*. Each scenario was conducted in a controlled AWS environment to ensure safety and reproducibility. Logs generated from these tests were systematically collected, analyzed, and appended as supporting material, offering tangible insights into how malicious activities manifest in native AWS logging services. This empirical groundwork validates the effectiveness of the proposed Sigma-based detection patterns and highlights pathways for further refinement.

Ultimately, the conclusions of this research emphasize the interplay between AWS's robust security services and the inherent complexities of operating at scale in a cloud environment. While the service portfolio of AWS—especially in the realms of incident response and compliance—offers substantial protective measures, organizations must remain cognizant of the associated learning curves, costs, and intricate configurations. Simultaneously, the intricacy of mapping TTPs to discrete actions underscores the dynamic nature of cloud-based threats, advocating for detection strategies rooted in flexible logic, thorough rule development, and continuous improvement. By combining in-depth analyses of AWS services with practical guidelines for implementing detection patterns, this thesis aspires to serve as a valuable resource for security practitioners aiming to fortify their cloud deployments against an evolving threat landscape.

REFERENCES

- [1] RAMAKRISHNAN, Ganesh and HAQANEE, Mansoor, 2024. *Cloud Forensics Demystified: Decoding Cloud Investigation Complexities for Digital Forensic Professionals*. Packt Publishing. ISBN 1800564414.
- [2] CLINTON, David and PIPER, Ben, 2023. *AWS Certified Cloud Practitioner Study Guide With 500 Practice Test Questions: Foundational (CLF-C02) Exam (Sybex Study Guide)*. Packt Publishing. ISBN 1394235631.
- [3] LAT, Joshua Arvin, 2022. *Machine Learning Engineering on AWS*. Packt Publishing. ISBN 9781803231389.
- [4] BOOK, Adam and SCOTT, Stuart, 2024. *AWS Certified Security - Specialty (SCS-C02) Exam Guide*. Packt Publishing.
- [5] GUPTA, C. P. and GOYAL, K. K., 2020. *Cybersecurity: A Self-Teaching Introduction*. Packt Publishing.
- [6] SERVICES, Amazon Web, 2023. AWS Identity and Access Management Documentation. Available from: <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>.
- [7] SERVICES, Amazon Web, 2023. AWS IAM Identity Center (AWS Single Sign-On) Documentation. Available from: <https://docs.aws.amazon.com/singlesignon/latest/userguide/what-is.html>.
- [8] SERVICES, Amazon Web, 2025. AWS Secrets Manager User Guide. Available from: <https://docs.aws.amazon.com/pdfs/secretsmanager/latest/userguide/secretsmanager-userguide.pdf#intro>.
- [9] SERVICES, Amazon Web, 2025. AWS Secrets Manager API Reference. Available from: <https://docs.aws.amazon.com/pdfs/secretsmanager/latest/apireference/secretsmanager-apireference.pdf>.
- [10] SERVICES, Amazon Web, 2025. AWS WAF Developer Guide. Available from: <https://docs.aws.amazon.com/pdfs/waf/latest/developerguide/waf-dg.pdf>.

- [11] SERVICES, Amazon Web, 2025. AWS WAF API Reference. Available from: <https://docs.aws.amazon.com/pdfs/waf/latest/APIReference/waf-apiref.pdf>.
- [12] SERVICES, Amazon Web, 2025. Amazon VPC Security Groups Overview. Available from: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html.
- [13] SERVICES, Amazon Web, 2025. AWS Security Hub. Available from: <https://docs.aws.amazon.com/pdfs/securityhub/latest/userguide/securityhub.pdf#what-is-securityhub>.
- [14] RAJAGOPAL, Jagan, 2024. *Understanding AWS Security Hub and Its Business Use Cases*. Online. Available from: <https://awstrainingwithjagan.com/understanding-aws-security-hub-and-its-business-use-cases/>.
- [15] SERVICES, Amazon Web, 2025. Amazon GuardDuty User Guide. Available from: <https://docs.aws.amazon.com/pdfs/guarddduty/latest/ug/guarddduty-ug.pdf>.
- [16] SERVICES, Amazon Web, 2025. AWS CloudTrail User Guide. Available from: <https://docs.aws.amazon.com/pdfs/awsccloudtrail/latest/userguide/awsccloudtrail-ug.pdf#cloudtrail-user-guide>.
- [17] *Audit API calls*. Online. Available from: <https://maturitymodel.security.aws.dev/en/1.-quickwins/cloudtrail/>.
- [18] AMAZON WEB SERVICES, Inc., 2025. Amazon Cognito. Available from: <https://docs.aws.amazon.com/pdfs/cognito/latest/developerguide/cognito-dg.pdf#cognito-user-identity-pools>.
- [19] CHIERICI, Stefano. Exploiting IAM security misconfigurations. Available from: <https://sysdig.com/blog/iam-security-misconfiguration/>.
- [20] AMAZON WEB SERVICES, Inc., . AWS Directory Service. Available from: <https://docs.aws.amazon.com/pdfs/directoryservice/latest/admin-guide/directoryservice-admin-guide.pdf>.
- [21] AMAZON WEB SERVICES, Inc., . AWS Compliance Program. Available from: <https://aws.amazon.com/compliance/programs/>.

- [22] 2025. Microsoft Entra ID Pricing. Available from: <https://azure.microsoft.com/en-us/pricing/details/microsoft-entra-external-id/>.
- [23] . AWS IAM Identity Center. Available from: <https://aws.amazon.com/iam/identity-center/>.
- [24] Amazon GuardDuty pricing. Available from: <https://aws.amazon.com/guardduty/pricing/>.
- [25] Microsoft Defender for Cloud pricing. Available from: <https://azure.microsoft.com/en-us/pricing/details/defender-for-cloud/>.
- [26] 2025. AWS Config. Available from: <https://docs.aws.amazon.com/pdfs/config/latest/developerguide/config-dg.pdf>.
- [27] aws.amazon.com. AWS Audit Manager. Available from: <https://docs.aws.amazon.com/pdfs/audit-manager/latest/userguide/user-guide.pdf.pdf#what-is>.
- [28] 2025. AWS Artifact. Available from: <https://aws.amazon.com/artifact/>.
- [29] 2025. AWS Config Pricing. Available from: <https://aws.amazon.com/config/pricing/>.
- [30] 2025. AWS Audit Manager pricing. Available from: <https://aws.amazon.com/>.
- [31] 2025. Azure Policy pricing. Available from: <https://azure.microsoft.com/en-us/pricing/details/azure-policy/>.
- [32] MEGAN RODDIE, Jason Deyalsingh and KATZ, Gary J., 2023. *Practical Threat Detection Engineering*. Packt Publishing. ISBN 1801076715.
- [33] *What is the Pyramid of Pain?*, 2019. Online. Available from: <https://www.attackiq.com/glossary/pyramid-of-pain/>.
- [34] MITRE ATT&CK for Enterprise - Cloud Matrix. Available from: <https://attack.mitre.org/matrices/enterprise/cloud/>.
- [35] *Ke3chang*. Online. Available from: <https://attack.mitre.org/groups/G0004/>.

- [36] *AWS Secrets Manager*, 2024. Online. Available from: <https://docs.aws.amazon.com/pdfs/secretsmanager/latest/userguide/secretsmanager-userguide.pdf#intro>.
- [37] About Sigma. Available from: <https://sigmahq.io/docs/guide/about>.
- [38] SERVICES, Amazon Web, 2025. Boto3 Documentation. Available from: <https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>.
- [39] SERVICES, Amazon Web, . Best Practices for the AWS Account Root User. Available from: <https://docs.aws.amazon.com/IAM/latest/UserGuide/root-user-best-practices.html>.
- [40] RIANCHO, Andres, 2023. Enumerate IAM. Available from: <https://github.com/andresriancho/enumerate-iam>.
- [41] DOCUMENTATION, AWS, 2025. DescribeFpgaImageAttribute. Available from: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeFpgaImageAttribute.html.
- [42] DOCUMENTATION, AWS, 2025. GetInstanceTpmEkPub. Available from: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_GetInstanceTpmEkPub.html.
- [43] DOCUMENTATION, AWS, 2025. GetCoipPoolUsage. Available from: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_GetCoipPoolUsage.html.
- [44] CROWDSTRIKE, 2023. How Adversaries Persist with AWS User Federation. Available from: <https://www.crowdstrike.com/en-us/blog/how-adversaries-persist-with-aws-user-federation/>.
- [45] BLOG, AWS Security, 2022. How to Revoke Federated Users' Active AWS Sessions. Available from: <https://aws.amazon.com/blogs/security/how-to-revoke-federated-users-active-aws-sessions/>.
- [46] DOCUMENTATION, AWS. GetFederationToken. Available from: https://docs.aws.amazon.com/STS/latest/APIReference/API_GetFederationToken.html.

- [47] AMAZON WEB SERVICES, Inc., . CreateToken - AWS IAM Identity Center. Available from: https://docs.aws.amazon.com/singlesignon/latest/OIDCAPIReference/API_CreateToken.html.
- [48] BERGER, Lex. Anatomy of AWS SSO Device Authorization Grant. Available from: <https://medium.com/@lex.berger/anatomy-of-aws-sso-device-authorization-grant-2839008c367a>.
- [49] AMAZON WEB SERVICES, Inc., . GetRoleCredentials - AWS IAM Identity Center. Available from: https://docs.aws.amazon.com/singlesignon/latest/PortalAPIReference/API_GetRoleCredentials.html.
- [50] . AssumeRoleWithSAML - AWS Security Token Service. Available from: https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithSAML.html.
- [51] BLAŽEK, Jan, 2023. Analysis and Detection of PWS Malware. Advised by Ing. Petr Dzurenda, Ph.D.
- [52] DAGRZ. aws_pwn. Available from: https://github.com/dagrz/aws_pwn.
- [53] LABS, Rhino Security, 2023. Pacu: The AWS Exploitation Framework. Available from: <https://github.com/RhinoSecurityLabs/pacu>.
- [54] SECURITY, Truffle. TruffleHog. Available from: <https://github.com/trufflesecurity/trufflehog/tree/3ee9c2f1af6365e0db6dc036826196bae21d34e7>.
- [55] BLOG, OmegaPoint Security. Write-up: TruffleHog. Available from: <https://securityblog.omegapoint.se/en/writeup-trufflehog/>.
- [56] CYBERARK. Golden SAML: Newly Discovered Attack Technique Forges Authentication to Cloud Apps. Available from: <https://www.cyberark.com/resources/threat-research-blog/golden-saml-newly-discovered-attack-technique-forges-authentication-to-cloud-apps>.
- [57] SERVICES, Amazon Web, . AWS Well-Architected Framework - Security Pillar. Available from: <https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html>.

- [58] SERVICES, Amazon Web, . AWS Config - Restricted Common Ports.
Available from: <https://docs.aws.amazon.com/config/latest/developerguide/restricted-common-ports.html>.
- [59] TEAM, Stratus Red. AWS Credential Access: EC2 Get Password Data.
Available from: <https://stratus-red-team.cloud/attack-techniques/AWS/aws.credential-access.ec2-get-password-data/>.
- [60] SPLUNK. Anomaly Detection. Available from: https://www.splunk.com/en_us/blog/learn/anomaly-detection.html.

LIST OF FIGURES

Figure 3.1	AWS Security Hub Integration With Other Services (from [14])	34
Figure 3.2	AWS CloudTrail (from [17])	39
Figure 5.1	Pyramid of Pain (from [33])	60

LIST OF LISTINGS

Listing 3.1	Restrictive IAM Policy Example	28
Listing 3.2	Custom WAF Rule Example	31
Listing 3.3	Custom Insight Example	34
Listing 5.1	Example of CreateUser event in AWS CloudTrail logs	61
Listing 5.2	Example of PutTargets API call in AWS EventBridge logs	62
Listing 7.1	Sigma rule identifying IAM calls that may be used for persistence	72
Listing 7.2	Sigma rule identifying successful AWS console sign-in	72
Listing 7.3	Sigma rule identifying API calls executed by automation scripts	73
Listing 7.4	Detection Rule of Root Login Without MFA	74
Listing 7.5	Root Login Without MFA Converted Into Splunk Query	74
Listing 7.6	Enumerate-iam Tool Signature (Unusual Payload) detection rule	75
Listing 7.7	Enumerate-iam Tool Signature (Unusual API Calls Sequence) detection rule	77
Listing 7.8	Enumerate-iam Tool Signature (Unusual API Calls Sequence) Converted Into Splunk Query	77
Listing 7.9	Enumerate-iam Tool Signature (Unusual Payload) Converted Into Splunk Query	78
Listing 7.10	Sigma detection rule identifying operations performed by federated users	79
Listing 7.11	Sigma detection rule identifying federated user logging into the AWS console	80
Listing 7.12	Sigma detection rule identifying federated user performing IAM persistence	80
Listing 7.13	AWS CLI SSO Cache Data	83
Listing 7.14	AWS CLI SSO Cache Data	83
Listing 7.15	Sigma detection rule identifying possible SSO access key abuse	85
Listing 7.16	SSO Access Key Theft Converted Into Splunk Query Language	85
Listing 7.17	Event Pattern Processing CreateRole IAM Function	87
Listing 7.18	Example PutTargets Request Parameters	88
Listing 7.19	Sigma Detection Rule for Lambda Backdoor	88

Listing 7.20	Splunk Query Identifying Lambda Backdoor	90
Listing 7.21	TruffleHog Code Creating User Agent	91
Listing 7.22	Sigma Detection Rule for TruffleHog	91
Listing 7.23	TruffleHog Detection Rule Converted Into ESQL	92
Listing 7.24	Successful SAML Provider Update Sigma Rule	93
Listing 7.25	Successful SAML Provider Update Rule Converted Into Splunk Query .	93
Listing 7.26	Sigma Rule for Pacu Detection Enumeration Module	95
Listing 7.27	Sigma Rule for Pacu Trail Minimization	96
Listing 7.28	Sigma Detection Rule for Allowing Unrestricted SSH Ingress Traffic	97
Listing 7.29	Unrestricted Ingress Access to Sensitive Ports Splunk Query	98
Listing 7.30	Sigma Detection Rule for Allowing Unrestricted SSH Ingress Traffic	99
Listing 0.31	Python script for resolving nested rule references.	115
Listing 0.32	Detection	117
Listing 0.33	Child Sigma Rules for Pacu Detection Enumeration Module	118
Listing 0.34	Sigma Detection Rules for Allowing Unrestricted RDP/SMB Ingress Traffic	119

LIST OF ABBREVIATIONS

ACL	Access Control List
ACM	Amazon Certificate Manager
API	Application Programming Interface
AWS	Amazon Web Services
C2	Command and Control
CI/CD	Continuous Integration and Continuous Deployment
CLI	Command Line Interface
COIP	Customer-Owner IP
CRM	Customer Relationship Management
DDoS	Distributed Denial of Service
ERP	Enterprise Resource Planning
ESQL	Elastic Search Query Language
FPGA	Field-Programmable Gate Array
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IAM	Identity and Access Management
IT	Information Security
KMS	Key Management Service
MFA	Multi Factor Authentication
PaaS	Platform as a Service
RDP	Remote Desktop Protocol
SAML	Security Assertion Markup Language
SaaS	Software as a Service
SIEM	Security Information and Event Management
SSH	Secure Shell
SSO	Single Sign-On
STS	Secure Token Service
TPM	Trusted Platform Module
TTP	Tactics, Techniques, and Procedure
VPC	Virtual Private Cloud
WAF	Web Application Firewall
WORM	Enforce-Write-Once-Read-Many
XSS	Cross-Site Scripting

LIST OF APPENDICES

APPENDIX A I: Sigma Reference Resolver

APPENDIX A II: Detection Rules Related to Lambda Backdoor

APPENDIX A III: Pacu Detection Enum Module Rules

APPENDIX A IV: Sigma Detection Rule for Unrestricted Ingress Port Access

APPENDIX A V: Content of the Electronic Attachment

APPENDIX A I: SIGMA REFERENCE RESOLVER

```
1 import yaml
2 import sys
3 import os
4 from copy import deepcopy
5
6 def load_rules(file_path):
7     with open(file_path, 'r') as f:
8         return list(yaml.safe_load_all(f))
9
10 def find_rule_by_id_or_name(rules, ref):
11     for rule in rules:
12         if rule.get('id') == ref or rule.get('name') == ref:
13             return rule
14     return None
15
16 def resolve_references(rules):
17     resolved_rules = []
18     rule_dict = {rule.get('id', rule.get('name')): rule for
19                  rule in rules}
20     for rule in rules:
21         detection = rule.get('detection', {})
22         condition = detection.get('condition', '')
23         if not condition:
24             resolved_rules.append(rule)
25             continue
26
27         tokens = condition.split()
28         new_condition = condition
29
30         for token in tokens:
31             ref_rule = find_rule_by_id_or_name(rules, token)
32             if ref_rule:
33                 ref_id = ref_rule.get('id', ref_rule.get('name'))
34                 prefixed_selection_name = f"{ref_id}
35                                         __selection"
36                 if prefixed_selection_name not in detection:
37                     ref_selection = ref_rule.get('detection',
38                                                  {}).get('selection', {})
39                     detection[prefixed_selection_name] =
40                         deepcopy(ref_selection)
41
42                 new_condition = new_condition.replace(token,
43                                                         prefixed_selection_name)
44
45         detection['condition'] = new_condition
46         rule['detection'] = detection
47         resolved_rules.append(rule)
48     return resolved_rules
49
50 def save_rules(rules, output_path):
51     with open(output_path, 'w') as f:
52         yaml.dump_all(rules, f, sort_keys=False)
53
54 def main():
55     if len(sys.argv) != 3:
56         print("Usage: python resolve_sigma_references.py <
57               input_rules.yaml> <output_resolved_rules.yaml>")
58         sys.exit(1)
59
60     input_file = sys.argv[1]
61     output_file = sys.argv[2]
62
63     if not os.path.exists(input_file):
64         print(f"Error: Input file '{input_file}' does not
65               exist.")
66         sys.exit(1)
67
68     rules = load_rules(input_file)
69     resolved_rules = resolve_references(rules)
70     save_rules(resolved_rules, output_file)
71     print(f"Resolved rules saved to '{output_file}'.")
```

```
69 if __name__ == "__main__":  
70     main()
```

Listing 0.31 Python script for resolving nested rule references.

APPENDIX A II: DETECTION RULES RELATED TO LAMBDA BACK-DOOR

```
1 title: EC2 Event Rule Pattern
2 name: event_rule_pattern
3 logsource:
4   product: aws
5   service: cloudtrail
6 detection:
7   selection:
8     eventName: PutRule
9     eventSource: events.amazonaws.com
10    eventPattern|contains: "aws.ec2"
11    event_name_key:
12      eventPattern|contains: "eventName"
13    event_names:
14      eventPattern|contains: "CreateSecurityGroup"
15    condition: (selection and not event_name_key) or (selection
16      and event_names)
17 title: Event Targets Lambda
18 name: event_targets_lambda
19 logsource:
20   product: aws
21   service: cloudtrail
22 detection:
23   selection:
24     eventName: PutTargets
25     eventSource: events.amazonaws.com
26     requestParameters.targets.arn|contains: ":lambda:"
27   condition: selection
28 ---
29 title: Lambda Function Processing Suspicious EC2 Events
30 description: "Lambda functions can be configured to trigger
31   execution when a specific event occurs in an AWS
32   environment. This detection identifies a Lambda function
33   processing suspicious EC2 events. Threat actors have
34   previously been observed using Lambda functions to process
35   sensitive EC2 events, potentially backdooring newly
36   created entities."
37 author: Jan Blazek
38 creation-date: 2/8/2025
39 severity: High
40 confidence: Low
41 license: MIT
42 investigation:
43   - "Review the the activity and verify if the activity is
44     legitimate."
45   - "Inspect the code inside the lambda function and validate
46     that the code is not malicious"
47   - "Investigate how the lambda function was created in the
48     environment."
49 response:
50   - "Revert all changes performed by the lambda function."
51   - "Perform containment functions on the lambda function and
52     its related entities."
53   - "Remove the event rule."
54 logsource:
55   product: aws
56   service: cloudtrail
57 correlation:
58   type: temporal
59   rules:
60     - event_rule_pattern
61     - event_targets_lambda
62 aliases:
63   rule_name:
64     event_rule_pattern: requestParameters.name
65     event_targets_lambda: requestParameters.rule
66 group-by:
67   - rule_name
68 timespan: 7d
69 timespan: 7d
```

Listing 0.32 Detection

APPENDIX A III: PACU DETECTION ENUM MODULE RULES

```
1 title: AWS Describe Trails Event
2 name: describe_trails
3 logsource:
4   product: aws
5   service: cloudtrail
6 detection:
7   selection:
8     eventName: DescribeTrails
9     eventSource: cloudtrail.amazonaws.com
10  condition: selection and automated_execution
11 ---
12 title: AWS List Detectors Event
13 name: list_detectors
14 logsource:
15   product: aws
16   service: cloudtrail
17 detection:
18   selection:
19     eventName: ListDetectors
20     eventSource: guardduty.amazonaws.com
21  condition: selection and automated_execution
22 ---
23 title: AWS Describe Alarms Event
24 name: describe_alarms
25 logsource:
26   product: aws
27   service: cloudtrail
28 detection:
29   selection:
30     eventName: DescribeAlarms
31     eventSource: monitoring.amazonaws.com
32  condition: selection and automated_execution
33 ---
34 title: AWS Describe Flow Logs Event
35 name: describe_flow_logs
36 logsource:
37   product: aws
38   service: cloudtrail
39 detection:
40   selection:
41     eventName: DescribeFlowLogs
42     eventSource: ec2.amazonaws.com
43  condition: selection and automated_execution
44 ---
45 title: AWS Get Subscription State Event
46 name: get_subscription_state
47 logsource:
48   product: aws
49   service: cloudtrail
50 detection:
51   selection:
52     eventName: GetSubscriptionState
53     eventSource: shield.amazonaws.com
54  condition: selection and automated_execution
55 ---
56 title: AWS Config Enumeration Events
57 name: config_enumeration
58 logsource:
59   product: aws
60   service: cloudtrail
61 detection:
62   selection:
63     eventSource: config.amazonaws.com
64     eventName:
65       - DescribeConfigRules
66       - DescribeDeliveryChannels
67       - DescribeDeliveryChannelStatus
68       - DescribeConfigurationRecorders
69       - DescribeConfigurationRecorderStatus
70       - DescribeConfigurationAggregators
71  condition: selection and automated_execution
```

Listing 0.33 Child Sigma Rules for Pacu Detection Enumeration Module

APPENDIX A IV: SIGMA DETECTION RULE FOR UNRESTRICTED INGRESS PORT ACCESS

```
1 title: "Unrestricted RDP Inbound Traffic"
2 description: "This detection identifies an EC2 security group
   allowing inbound RDP traffic from any IP address."
3 author: Jan Blazek
4 creation-date: 09/29/2024
5 license: MIT
6 tags:
7   - attack.T1562
8 severity: Medium
9 investigation:
10  - "Verify the reason for this activity with the user who
    performed the operation."
11  - "Review SSH logs from the EC2 instance associated with
    the affected security group."
12 response: "Restore a secure security group configuration."
13 logsource:
14   product: aws
15   service: cloudtrail
16 detection:
17   selection:
18     eventName: AuthorizeSecurityGroupIngress
19     eventSource: ec2.amazonaws.com
20     requestParameters.fromPort|lte: 3389
21     requestParameters.toPort|gte: 3389
22     responseElements.requestId|contains: "-"
23   ipv4Unrestricted:
24     requestParameters.cidrIp|contains: 0.0.0.0
25   ipv6Unrestricted:
26     requestParameters.cidrIpv6: ::/0
27   condition: selection and (ipv6Unrestricted or
    ipv4Unrestricted)
28 ---
29 title: "Unrestricted SMB Inbound Traffic"
30 description: "This detection identifies an EC2 security group
   allowing inbound SMB traffic from any IP address."
31 author: Jan Blazek
32 creation-date: 09/29/2024
33 license: MIT
34 tags:
35   - attack.T1562
36 severity: Medium
37 investigation:
38  - "Verify the reason for this activity with the user who
    performed the operation."
39  - "Review SSH logs from the EC2 instance associated with
    the affected security group."
40 response: "Restore a secure security group configuration."
41 logsource:
42   product: aws
43   service: cloudtrail
44 detection:
45   selection:
46     eventName: AuthorizeSecurityGroupIngress
47     eventSource: ec2.amazonaws.com
48     requestParameters.fromPort|lte: 445
49     requestParameters.toPort|gte: 445
50     responseElements.requestId|contains: "-"
51   ipv4Unrestricted:
52     requestParameters.cidrIp|contains: 0.0.0.0
53   ipv6Unrestricted:
54     requestParameters.cidrIpv6: ::/0
55   condition: selection and (ipv6Unrestricted or
    ipv4Unrestricted)
```

Listing 0.34 Sigma Detection Rules for Allowing Unrestricted RDP/SMB Ingress Traffic

APPENDIX A V: CONTENT OF THE ELECTRONIC ATTACHMENT

The figure below illustrates the structure of the content in the electronic attachment. The root directory contains subfolders corresponding to each technique described in the thesis. Each subfolder includes the file *rule.yaml*, which contains one or more Sigma rules, the file *telemetry.json*, which holds the telemetry generated during testing, and the file *query.txt*, which provides the Splunk query derived from the Sigma rule.

```
/ ..... root directory of the attached archive
├── 1_root_sign_in_withou_mfa . folder related to specific technique with expanded content
│   ├── rule.yaml
│   ├── telemetry.json
│   └── query.txt
├── 2_enumerate_iam_tool
├── 3_federation_token_abuse
├── 4_sso_credential_theft
├── 5_lambda_backdoor
├── 6_trufflehog_abuse
├── 7_golden_saml
├── 8_pacu
├── 9_sensistive_ports
├── 10_high_volume_of_unusual_calls
└── rule_preprocessor.py ..... python script preprocessing the rules
```