National College of Ireland

Customized Penetration Testing Framework for Assessing the Security of Amazon Web Services (AWS)

MSc Research Project
MSc Cybersecurity

Piyush Nikam
21202931

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Piyush Sanjay Nikam
Student ID:	21202931
Programme:	MSc Cybersecurity **Year:** Sep 2022 -Sep 2023
Module:	MSc Research Project
Supervisor:	Michael Prior
Submission Due Date:	14th Aug 2023
Project Title:	Customized Penetration Testing Framework for Assessing the Security of Amazon Web Services (AWS)
Word Count:	6158 **Page Count** 21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Piyush Sanjay Nikam

Date: 14th Aug 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Customized Penetration Testing Framework for Assessing the Security of Amazon Web (AWS)

Piyush Nikam

21202931

Abstract

The objective of this research endeavor is to construct a personalized framework for conducting penetration tests, tailored to evaluate the security aspects of Amazon Web Services (AWS). The intended framework will possess the capability to pinpoint and rectify vulnerabilities prevalent in the extensively utilized AWS services. The research methodology encompasses multiple facets, including the judicious selection of pertinent AWS services, the utilization of established tools, and the implementation of the Python programming language to forge the framework. Delving into the specifics, the proposal outlines the various phases and steps characterizing the penetration testing procedure. Furthermore, ethical considerations inherent in the research undertaking will be thoughtfully addressed. In culmination, this research is poised to yield a pragmatic and efficacious penetration testing framework specially designed for AWS, thereby offering a valuable means to fortify cloud infrastructure against potential threats.

1 Introduction

The rapid and dynamic increase in the cloud computing landscape has made it a prominent subject of research. As organizations embrace benefits provided by cloud computing, but it is crucial to be aware of potential vulnerabilities that can compromise the confidentiality and integrity of the data stored within it (Rohith *et al.*, 2023). One of the most eminent cloud service providers, Amazon Web Services (AWS), offers a suite of solutions that boast scalability, flexibility, and cost-effectiveness. However, as several research studies have highlighted, the multifaceted offerings by AWS are not devoid of challenges. Notably, while certain studies emphasize the prevention of security attacks in cloud computing and propose hybrid cloud models (Gaur *et al.*, 2023), others provide overviews of security concerns in cloud computing, emphasizing the shared responsibility between cloud providers and customers in ensuring security (Reddy *et al.*, 2023). Another research work provided insights into level-wise classification of security issues in cloud computing, ranging from network to application levels, emphasizing the novel approach in analyzing these challenges (Chaudhari *et al.*, 2022).

This research seeks to dive deep into these challenges and vulnerabilities and aims to introduce a penetration testing framework tailored for AWS, which combines various methods and tools and automation to scrutinize AWS services. Utilizing a grey-box approach, this

research delves into the three core stages of the Vulnerability Assessment and Penetration Testing (VAPT) process: Scanning, Vulnerability Assessment, and Exploitation.

1.1 Motivation

The imperative nature of robust cloud security practices was underscored during a personal encounter with security vulnerabilities in my professional journey. Despite adhering to AWS security protocols, an unforeseen breach occurred due to a system misconfiguration, granting unauthorized access to an attacker. This breach was later discovered to be a concealed Bitcoin mining operation that spanned multiple AWS regions. The cost of this security oversight was approximately 8000 euros, making evident the critical gaps in the then-existing security strategy.

Several pieces of research concur with such real-world examples, highlighting the escalating risks associated with cloud computing. A notable IBM report from 2021 elucidated that the average time taken to detect and address a data breach stood at 287 days, with a staggering average cost of $4.24 million[1]. The same report detailed that the maturity stage of cloud development inversely correlates with the financial implications of a breach. In essence, organizations in the nascent stages of cloud development incur significantly higher breach costs.

1.2 Research Problem

Given the wide scope of cloud security and its intricacies, this research primarily focuses on the following question: How do tailored penetration testing methods compare to generic ones in identifying vulnerabilities within the AWS ecosystem?

To facilitate this investigation, a detailed framework has been meticulously constructed. This framework helps to discern the efficiency of specialized penetration testing techniques with standard methods, especially within the AWS realm. The foundation of this framework is rooted in a literature review that showcased varied cloud security challenges, ranging from securing authentication to mitigating malware injections (Rohith *et al.*, 2023) and security misconfigurations. Another body of work provided insights into level-wise classification of security issues in cloud computing, ranging from network to application levels, emphasizing the novel approach to analyzing these challenges (Chaudhari *et al.*, 2022).

As cloud computing solidifies its position as an integral facet of the IT industry, it is of paramount importance to ensure the holistic security of cloud-based applications and the invaluable data they harbor (Raj *et al.*, 2022). Through this research, the aim is to bridge the gap between the complexities of AWS services and comprehensive security measures, thereby safeguarding the integrity of data, bolstering confidence among organizations, and contributing to the vast reservoir of knowledge on cloud security (Mallisetty *et al.*, 2023; Blancaflor *et al.*, 2023).

2 Related Work

[1] https://newsroom.ibm.com/2021-07-28-IBM-Report-Cost-of-a-Data-Breach-Hits-Record-High-During-Pandemic

2.1 Challenges and Threats in Cloud Security

The evolution of cloud computing has revolutionized how businesses operate, offering scalability and cost-effectiveness. However, the growth has also brought about significant security challenges. Recent studies provide critical analysis of the security risks and concerns tied to cloud computing systems, such as data breaches and vulnerabilities exploited by hackers (Tiwari *et al.*, 2021). The analysis includes issues like data confidentiality, insecure interfaces, malicious insiders, and compliance with regulations (Sureshkumar and Baranidharan, 2021). Additionally, cloud-specific cyber-attacks like flooding attacks, DOS, and cryptography vulnerabilities have become more prevalent, demanding constant vigilance and the implementation of advanced security measures (Roy and Patil, 2023; Singh and Kumar, 2023). A comprehensive overview of these challenges, including threats and attacks over cloud security, can be found in research papers (Jansi Sophia Mary *et al.*, 2023). This body of literature offers insights into the need for robust security measures, the evolving nature of cyber threats, and the importance of customized penetration testing techniques.

2.2 Penetration Testing and Vulnerability Assessment in Cloud Security

Given the intricate web of vulnerabilities in cloud infrastructure, penetration testing and vulnerability assessment (VAPT) are essential tools for enhancing security. In particular, the use of automated tools and frameworks like Vulcan and Potassium has been developed to address potential security flaws (Yurtseven and Bagriyanik, 2020). Additionally, research papers propose methodologies for automating the penetration testing process in cloud applications (Casola *et al.*, 2018). Tools such as the Metasploit Framework, widely used for scanning and exploiting systems, are emphasized for their vital role in identifying weaknesses and testing security measures (Raj and Walia, 2020). Other automated techniques in vulnerability assessment involve open-source tools and frameworks like Nmap, Kali Linux, and Metasploit to conduct efficient and effective testing (Mohan *et al.*, 2022). Research by Yurtseven and Bagriyanik (2020) and Casola et al. (2018) has also been instrumental in enhancing vulnerability assessments and proposing automated penetration testing methodologies, respectively (Yurtseven and Bagriyanik, 2020).

2.3 Solutions and Advances in Cloud Security

To tackle the complexities of cloud security, various solutions and approaches have been explored. The literature highlights unified cloud security taxonomies and emphasizes trust-based security models and privacy-preserving solutions (Kumar and Goyal, 2019). There's also a focus on the role of identity management, physical management, firewalls, tokenization, and other measures for secure cloud computing (Paul and Aithal, 2019). Newer innovations include tools that detect privilege escalations in Identity and Access Management (IAM) configurations in cloud platforms, utilizing abstract modeling and reinforcement learning (Hu *et al.*, 2023). A multi-layered security approach, along with comprehensive models like the Cloud Computing Adoption Framework (CCAF), has been proposed to systematically adopt and apply cloud security principles (Chang and Ramachandran, 2016).

2.4 Cloud Security for Small and Medium-Sized Enterprises (SMEs)

While cloud computing offers various advantages to businesses of all sizes, the literature has especially addressed the adoption of cloud security by small and medium-sized enterprises (SMEs). Research identifies several factors that influence SMEs' adoption of cloud security, including perceived usefulness, trust, human competency, and top management support (Roy and Patil, 2023). The studies suggest that transparency and collaboration among management, staff, and service providers can help SMEs tackle scalability and interoperability issues. Nevertheless, the literature also underlines the need for more extensive research, particularly employing larger sample sizes and qualitative investigation methods to understand cloud security adoption in SMEs (Roy and Patil, 2023).

2.5 Comprehensive Analysis of Vulnerability and Threat Management

Cloud computing's immense popularity in the business world has led to a corresponding increase in security concerns, such as DOS attacks and cryptography vulnerabilities. Recognizing the need for robust security measures, scholars and industry experts have undertaken extensive research to examine the challenges and propose strategies for mitigating risks in cloud computing environments (Singh and Kumar, 2023).

Yurtseven and Bagriyanik's (2020) work has proved to be instrumental in proposing systematic approaches to enhancing vulnerability assessments in cloud ecosystems (Yurtseven and Bagriyanik, 2020). Similarly, Casola et al. (2018) provide an automated penetration testing methodology using the Multicloud Application Composition Model (MACM), signifying a more efficient process compared to traditional methods (Casola *et al.*, 2018). Meanwhile, research also shows that SMEs' adoption of cloud security depends on factors such as trust, perceived usefulness, management support, and policies (Roy and Patil, 2023).

Automated tools and techniques, like Nmap and Metasploit, are leveraged in Vulnerability Assessment and Penetration Testing (VAPT) to provide comprehensive insights into potential system vulnerabilities (Mohan *et al.*, 2022). The paper titled "A Study on Metasploit Framework: A Pen-Testing Tool" highlights the critical role of such tools in identifying vulnerabilities and enhancing cybersecurity (Raj and Walia, 2020). Furthermore, methodologies like that in the paper "Towards Automated Penetration Testing for Cloud Applications" demonstrate the potential automation in the penetration testing process, reinforcing the effectiveness of these methods in cloud applications (Casola *et al.*, 2018).

2.6 The Evolution of Penetration Testing Techniques and Cloud Security Management

Given the sophisticated nature of cyber threats, the role of penetration testing and vulnerability assessment in the cloud environment cannot be overstated. Innovative techniques like interactive grey-box penetration testing and the application of deep reinforcement learning have emerged (Hu *et al.*, 2023).

In the context of AWS, security considerations are multi-dimensional. IAM modeling has shown promising results in detecting privilege escalations, as seen in the work by (Hu *et al.*, 2023). Additionally, the integration of machine learning-based security solutions in large-scale cloud infrastructures signifies an exciting area for future research (Singh and Kumar, 2023).

Existing frameworks like Vulcan have been specifically developed to identify and address possible security flaws in cloud systems (Yurtseven and Bagriyanik, 2020). The Multicloud Application Composition Model (MACM) demonstrates an example of using automation to enable more efficient vulnerability assessments (Yurtseven and Bagriyanik, 2020).

2.7 Ensuring Security in the Adoption and Application of Cloud Computing

The need for integrated and holistic security approaches is vital in addressing the myriad challenges of cloud computing security. The Cloud Computing Adoption Framework (CCAF) offers a comprehensive model that stresses the importance of considering stakeholders and risk assessments (Chang and Ramachandran, 2016).

The research also emphasizes data protection, identity management, and physical management, underscoring the need for effective security controls and proper attention from both service providers and users (Paul and Aithal, 2019). Emphasizing the importance of trust-based security models and privacy-preserving solutions, the literature also focuses on filling existing gaps, offering a unified cloud security taxonomy, and providing valuable insights for researchers and industry professionals (Kumar and Goyal, 2019).

Consequently, the dynamic nature of cloud security and the complexity of the associated threats necessitate a multifaceted approach (Hakani and Mann, 2022). Combining traditional security protocols with cutting-edge methodologies like deep reinforcement learning and integrating an understanding of business needs with technological advances, will likely shape the future of secure cloud computing. The current body of literature offers a foundational understanding of these aspects and points the way to continuous innovation and development in cloud security management.

Cloud computing is undeniably at the forefront of modern technology, and its adoption shows no signs of slowing down. Within this booming industry, Amazon Web Services (AWS) dominates, holding a 33 percent market share (Writer, 2022). Yet, its popularity is a double-edged sword, making it a tantalizing target for cyber attackers. Consequently, security is paramount for organizations leveraging AWS. Recognizing this urgent need, this study introduces a customized penetration testing framework designed specifically for AWS. Its objective? To help companies unearth hidden security vulnerabilities and potential threats within their AWS architecture.

While AWS has proven to be susceptible to attacks, its choice as the subject of this research is no mere coincidence. A comparative analysis of leading cloud platforms—including Amazon, Microsoft, and Google—revealed AWS's market share dropped by 1% in 2020. This serves as an alarming call to action for enhancing security measures to maintain its leadership position.

3 Research Methodology

3.1 Selection of AWS services for Penetration Testing Framework

In the highly complex environment of AWS, numerous services are commonly employed by organizations. Among them, some are more frequently targeted by cyber adversaries, owing to their prevalent use and integral nature within AWS infrastructures. In this research, a thoughtful and informed selection was made to identify the services that are at higher risk and hold significant importance to organizations.

Specifically, IAM (Identity and Access Management), EC2 (Elastic Compute Cloud), S3 (Simple Storage Service), and Lambda were chosen. These services, recognized for their vital roles in AWS operations, often become attractive targets for malicious activities. IAM, responsible for controlling access within the AWS environment, can be a critical point of exploitation if not properly configured. EC2(Mansour *et al.*, 2017), as a scalable compute capacity, handles significant portions of an organization's operations. S3(Latha *et al.*, 2022), known for storage solutions, and Lambda, with its serverless compute service, are also crucial. The selection process was a critical phase that ensured that the framework was not just a generic solution but a targeted instrument to assess and enhance AWS security.

3.2 Selection of Tools and Language

Creating a customized framework for penetration testing AWS services required meticulous planning and careful selection of existing tools and programming languages. The objective was to leverage proven technologies while adding unique functionalities tailored to AWS's specific requirements.

The research turned to Pacu; an existing penetration testing tool widely recognized for its effectiveness in AWS environments. Pacu's robustness and specialized functionalities provided a strong base upon which additional features could be built. Its compatibility with AWS made it an obvious choice, ensuring that the customized framework would be grounded in proven methodologies while still allowing for innovation. Python, known for its versatility and rich library ecosystem, was chosen as the development language (Abirami *et al.*, 2020; Muralidharan *et al.*, 2023). The decision was guided by Python's ability to integrate seamlessly with existing tools like Pacu, along with its flexibility to allow for the development of specialized modules targeting AWS services like IAM, EC2, S3, and Lambda. The alignment of these components was not a mere coincidental choice but a strategic decision, reflecting an in-depth understanding of the penetration testing landscape and AWS's specific vulnerabilities.

3.3 Creating AWS Infrastructure

The creation of a security misconfigured environment within AWS required a deep investigation into various tools capable of replicating real-world vulnerabilities. Two tools emerged as potential candidates for this purpose: AWSGoat by INE[2] and CloudGoat by Rhino Security Labs[3].

[2] https://ine.com/blog/awsgoat-a-damn-vulnerable-aws-infrastructure?utm_source=twitter&utm_medium=organic&utm_campaign=AWSGoatInfrastructure&utm_content=blog

[3] https://rhinosecuritylabs.com/aws/introducing-cloudgoat-2/

AWSGoat, a prominent tool, was explored in depth for its potential applicability to this research. However, after careful evaluation, it became apparent that AWSGoat's primary focus lay in identifying vulnerabilities within applications hosted on AWS, rather than the AWS services themselves.

Conversely, CloudGoat presented a more tailored solution for this research's objectives. Specifically designed to create complex interconnected AWS environments with deliberate security misconfigurations, CloudGoat offered a rich variety of AWS security misconfiguration scenarios. These scenarios encapsulate core AWS services and the intricacies of their potential weaknesses. It's not just about simulating vulnerabilities; it's about replicating the multifaceted reality of AWS's security landscape in a controlled environment.

The decision to utilize CloudGoat was backed by a methodical assessment of its capabilities and alignment with the research's overarching goals. Through its nuanced simulation of AWS service misconfigurations, CloudGoat proved to be an invaluable asset in developing the customized penetration framework for AWS.

3.4 Developing the Framework for Pentesting AWS Services

Developing a robust and flexible penetration testing framework required a well-thought-out design methodology. Inspired by widely used penetration frameworks like Metasploit, a modular approach was adopted. This approach not only aligns with the best practices in software development but also provides a path for the framework to evolve in tandem with the ever-changing threat landscape.

In this modular design, each module encapsulates specific functionality, allowing for better organization, separation of concerns, and adaptability. Like building blocks, modules can be arranged, modified, or replaced without affecting the core framework. This structure fosters contribution, maintenance, and extension with new capabilities.

The framework's focus remains anchored to the latest security misconfigurations and vulnerabilities discovered in AWS. Leveraging CloudGoat's different scenarios, it has been designed to swiftly detect and remediate these vulnerabilities. It’s not a static tool but a dynamic solution, capable of evolving with the threats it seeks to mitigate. By ensuring that the framework can be easily modified, it offers a responsive and adaptive shield against AWS vulnerabilities. The development process highlighted the importance of design principles, like modularity, that doesn’t just respond to current challenges but anticipate future developments.

3.5 Testing the Framework on the AWS Infrastructure

The culmination of this research's methodological journey was the application and testing of the developed framework on the AWS infrastructure. An essential phase, this testing sought to evaluate the efficacy of the individual modules in detecting and diagnosing security misconfigurations within the simulated environment.

The AWS infrastructure for this exercise was created in step 3, utilizing CloudGoat to craft a complex, interconnected landscape replete with deliberate security misconfigurations. This environment was not a mere academic construct; it was a virtualized replica of real-world AWS scenarios, reflecting the intricate web of potential vulnerabilities that organizations may face.

The testing procedure was systematic and rigorous, designed to probe the framework's capabilities in both speed and precision. With a focus on evaluating whether the framework could quickly identify security misconfigurations. Automation played a crucial role here, with predefined steps guided by the specific characteristics of each detected misconfiguration. The testing was not a broad sweep but a targeted exploration, each step aligned with the nuanced landscape of AWS security.

The modular approach enabled fine-tuning and adjustments as required, reaffirming the wisdom of the design choice. Also, the success of this testing phase did not merely validate the technical soundness of the framework. It affirmed the methodological robustness of the entire research process. From the careful selection of AWS services to the thoughtful design of the framework and the rigorous simulation of the AWS environment, every step was validated through this testing.

3.6 Case Scenarios

The assessment of the customized penetration framework's effectiveness would be incomplete without practical application through various real-world case scenarios. These scenarios, crafted using CloudGoat and additional manual configurations, provided an essential proving ground for the framework's capabilities. Below, I will detail three key scenarios that formed the centerpiece of this research:

3.6.1 Scenario 1: Scanning IAM Permissions

The first scenario focused on scanning the AWS Identity and Access Management (IAM) permissions. Different IAM services were set up within AWS, and the customized framework was employed to perform a comprehensive scan. The objective was to discern the framework's ability to find information related to permissions and assess the IAM services' security posture. This scenario emphasized how vital understanding and evaluating permissions within AWS is, as misconfigurations in IAM can lead to unauthorized access. The results obtained from this scenario offered a detailed glimpse into the potential vulnerabilities within IAM and the efficacy of the customized framework in identifying them.

3.6.2 Scenario 2: Elevating IAM User Access

In the second scenario, an IAM user was created with limited access rights. The initial attempt to gain AWS Administrator access through the AWS console was unsuccessful, mirroring real-world constraints. However, by employing the customized framework along with the integration of the Pacu Framework, an escalation to administrator access was achieved. This scenario demonstrated the framework's ability to work in conjunction with other penetration tools and the importance of a layered approach to security testing. It emphasized the complex nature of access controls within AWS and the need for sophisticated tools to unravel potential weaknesses.

3.6.3 Scenario 3: Exploiting Chained Security Misconfigurations

The third scenario was a multifaceted one, involving IAM, EC2, Lambda, and S3 services. The task was to exploit chained security misconfigurations to invoke a Lambda function that could

typically only be accessed by a high-privileged user. This scenario presented a more complex landscape, reflecting the intricate interdependencies within AWS services. It underscored the importance of understanding how different services can be chained together to form a security risk. The success in invoking the restricted Lambda function highlighted the customized framework's ability to navigate and exploit multi-layered security misconfigurations. These scenarios are more than mere exercises; they are reflective instances of what organizations might encounter in real-world AWS environments. Through a methodical approach, they provide valuable insights into potential vulnerabilities and affirm the strength, adaptability, and relevance of the customized framework. They resonate beyond the confines of academic research, offering practical lessons and direction for professionals concerned with AWS security. With rigorous design, implementation, and analysis, these scenarios contribute to the broader narrative of cybersecurity in the era of cloud computing.

4 Design Specifications

The important part of this research revolves around a carefully constructed framework named `AWSNeo`, designed to interact with AWS services such as IAM, EC2, Lambda, and S3. AWSNeo's architecture enables the exploration and exploitation of security loopholes within a controlled environment, providing invaluable insights into cloud security.

4.1 Architecture and Directory Structure

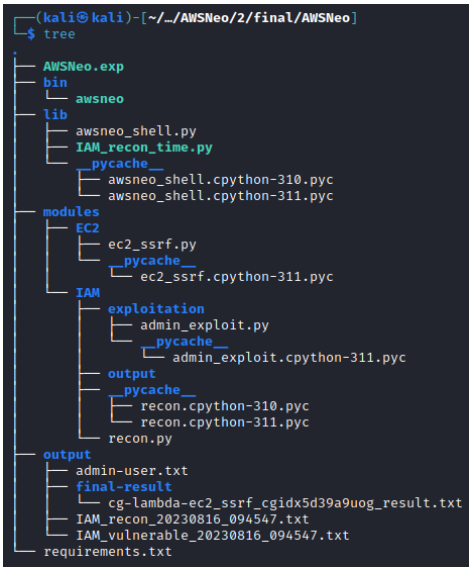

Figure 1:AWSNeo Directory Structure

The framework follows a modular and concise architecture, divided into three main directories: `bin`, `lib`, and `modules`. Each plays a significant role:

- bin: This contains the main executable for AWSNeo, which acts as the entry point to the framework.
- lib: This directory houses the core-shell of AWSNeo (`awsneo_shell.py`), which serves as the command interface. It also includes a cache for Python-compiled scripts.

- modules: The important part of AWSNeo's functionality, containing specific directories for different AWS services.
- Additionally, there are directories like `output`, where the results are stored, and a `requirements.txt` file that lists all the dependencies required for the framework.

4.2 AWSNeo

Main Shell of Functionality of AWSNeo('awsneo_shell.py')

The core of the framework resides in `awsneo_shell.py`. Here's a table of its major functionalities:

No	Name	Description
1	AWSNeoShell Class	The class provides an interactive shell, introducing the sections and commands available to the user.
2	AWS Credentials Setup	It allows users to set up AWS access and secret keys, facilitating interaction with AWS services.
3	IAM Recon Module	This is used to run IAM reconnaissance, integrating functionalities like exploitation and scanning.
4	IAM Exploit Module	A robust section utilizing the Pacu Framework to handle IAM privilege escalation scanning.
5	EC2 SSRF Module	A complex module designed to exploit SSRF vulnerabilities within EC2 instances, S3, Lambda and IAM services.

Table 1: AWSNeo Functionalities

The modular design not only improves maintainability but also enables a scalable structure. This is vital for future adaptations and expansions.

4.3 Kali OS on VirtualBox

The implementation leverages Kali Linux OS (version 2022.3) running on VirtualBox (version 7), optimized for running various tools, AWSNeo and creating AWS infrastructure. Kali Linux, part of the Debian family, was selected for its renowned performance in the security domain and compatibility with the required Linux-based tools (Tigner *et al.*, 2021). It is meticulously configured with an Internal Networking setup to isolate the Virtual Machines (VMs) from the host machine and the internet (Raceanu and Marian, 2023). The networking method utilized in the project was Bridged Adapter within VirtualBox, enabling a seamless connection while maintaining a secure environment.

4.4 CloudGoat

CloudGoat plays a critical role in simulating realistic AWS environments. The prerequisites for this module include:

1. Operating System Compatibility: Only Linux or MacOS were considered as Windows is not officially supported. Bash 4.2+ was required for argument tab completion.

- Python Requirements: Python 3.6 or above to ensure compatibility.
- Terraform: Version 0.14 or higher, installed and configured in the system's `$PATH`.
- AWS CLI: Installed in the `$PATH`, coupled with an AWS account with sufficient administrative permissions to create and destroy resources.
- Additional Tools: `jq` for processing JSON inputs.
- AWS IAM User: Specific administrative permissions were allocated for AWS infrastructure creation.

4.5 Pacu[4]

Pacu was utilized as a powerful AWS exploitation framework. It demanded specific prerequisites:

- Python Requirements: Python 3.7 or higher was necessary, along with pip3 to install various Python libraries. These libraries enable a robust connection with AWS services and enable the framework to execute complex queries and commands.
- Environment Customization: Adaptations were made to suit the project's needs, including custom configurations that align with the specific requirements of the AWS exploitation and security assessments.

The design specifications articulated above encapsulate the fundamental technologies and frameworks that underpin the project's implementation. By leveraging Kali OS on VirtualBox, CloudGoat, and Pacu, the system is architected to simulate, analyze, and exploit AWS environments in a controlled and secure manner.

4.6 AWS Account Configuration

To create and manipulate AWS infrastructure, a personal AWS account was set up explicitly for this research project. Since high-level privileges were a critical requirement to establish and control the diverse components of the infrastructure (Ksiezopolski *et al.*, 2022), the account was configured in accordance with specific AWS guidelines.

This enabled full control over the resources and facilitated the execution of administrative tasks vital to the research. By adhering to AWS's best practices, the account was optimized to ensure security, scalability, and efficiency, thereby aligning with the project's objectives.

Utilizing a dedicated AWS account also provided a controlled environment, allowing for precise monitoring and management of resources. This proved to be an invaluable asset in exploring and simulating various AWS-related scenarios within the research.

5 Implementation

5.1 Creating the AWS Infrastructure

The creation of the AWS infrastructure stood as a cornerstone in the research and was achieved through two distinct methodologies.

[4] https://rhinosecuritylabs.com/aws/pacu-open-source-aws-exploitation-framework/

5.1.1 Manual Creation of AWS Infrastructure

To test IAM exploits, a specific user dubbed "NCI" was meticulously created with circumscribed IAM service permissions. A custom policy was adjoined to this user. This manual process allowed for precise control over permissions and policy assignments, facilitating the focused examination of IAM-related vulnerabilities.

5.1.2 Automation for creating the AWS Infrastructure

To augment the research with more complex scenarios, the CloudGoat framework by Rhino Security Labs was leveraged. This required the creation of an administrator user "NCI_Cloud" given the necessity for higher-level permissions. The utilization of CloudGoat allowed for the orchestration of an interconnected ecosystem comprising services like EC2, Lambda, S3, and IAM. The specific scenario featured a Node.js app with an SSRF vulnerability hosted on EC2. This complex setup required careful coordination and involved a series of steps executed on Kali Linux running in VirtualBox:

Once prepared, an AWS infrastructure was adeptly constructed for the `ec2_ssrf` scenario, the complete details of which are shown in the accompanying screenshots. Additionally, the design allowed for the seamless deletion of the created AWS infrastructure using a single command, providing an efficient way to conclude each test iteration.

5.2 Pacu Tool configuration

The Pacu tool configuration was another pivotal aspect of the research, ensuring the smooth integration of IAM exploitation testing within our framework. The setup procedure of Pacu was straightforward, adhering to its official installation manual. The initial step was the acquisition of the tool by cloning its git repository. Subsequent installation steps were executed.

However, a mere installation wasn't sufficient for our objectives. Customizations were made to the `main.py` and `iam__privesc_scan` modules within Pacu. The intent behind these modifications was to ensure seamless integration with the AWSNeo framework. This integration allowed Asner to operate the Pacu `iam__privesc_scan` module as a subprocess, a feature that proved invaluable during our IAM admin exploitation scenario. This implementation not only ensured a structured approach to IAM exploitation but also showcased the versatility of combining standalone tools like Pacu within larger frameworks for enhanced functionality.

5.3 AWSNeo Modules

5.3.1 IAM Recon Module

Functionality:

- Initialization: The module is initiated via the `do_IAM_recon` function in `awsneo_shell.py`, ensuring that AWS credentials are in place before launching.
- IAM Client Configuration: Using the powerful boto3 library, an IAM client is initialized. This step sets the stage for all subsequent interactions with AWS IAM.

- Output Preparation: By generating a unique timestamp, the module ensures that each reconnaissance session's output is uniquely saved, minimizing data overlap.
- Data Gathering: Here lies the heart of the module. It fetches critical data, like:
  - Users: Each user's detailed profile, including policies and permissions, is compiled.
  - Roles: Each IAM role is scrutinized just as thoroughly.
- Visual Feedback: An animated ellipsis (three dots) offers users a live feedback mechanism, assuring them that the module is actively processing.
- Output Presentation: Post-processing, the path to the data's saved location is promptly displayed.

Significance:

For security assessments, IAM_recon is an invaluable tool. Whether you're conducting a standard security audit, verifying compliance, or orchestrating a penetration test, the information gathered by IAM_recon can unveil critical insights. Its incorporation into automated pipelines can further enable ongoing monitoring, thus amplifying security readiness.

5.3.2 IAM_exploit Module

Where IAM_recon is about gathering intelligence, IAM_exploit focuses on action. It leverages the information previously acquired to potentially exploit any vulnerabilities found within the AWS IAM framework.

Functionality:

- Credential Verification: Before any exploits can be attempted, AWS credentials need to be verified.
- Engaging with Pacu: By launching Pacu's CLI, the module bridges this AWS-centric exploitation framework, magnifying its capabilities.
- Leveraging iam__privesc_scan: This module from Pacu is invoked, likely to detect privilege escalation opportunities within IAM.
- Output Handling: The module smartly captures the command's raw output, cleans it for readability, and showcases it to the user.
- Process Termination: The child process used to run Pacu is responsibly terminated post-execution.

Significance:

IAM_exploit's true strength lies in its dual nature. On one hand, it's a robust testing tool, perfect for penetration testers and cybersecurity professionals. They can, with ease, detect and remedy weak IAM permissions.

5.3.3 ec2_ssrf Module:

SSRF (Server-Side Request Forgery) is a dangerous vulnerability that can be exploited to trick a server into making unintended requests. Within AWS, especially in EC2 instances, SSRF can reveal the instance's metadata, potentially leaking AWS IAM credentials. This is where the ec2_ssrf module comes into play.

Functionality:

- AWS Profile Configuration: Before any SSRF exploitation begins, a suitable AWS profile is set up.
- EC2 Instance Description: The module fetches information about active EC2 instances, pinpointing potential SSRF targets.
- SSRF Attack Walkthrough: Users are provided with a step-by-step guide to exploit SSRF, to tap into the EC2 instance's metadata.
- S3 Bucket Access: After acquiring credentials from the SSRF attack, users can now interact with associated S3 buckets.
- Lambda Functions Operations: Apart from S3 buckets, users can also invoke Lambda functions, either through the initial profile or the SSRF-obtained credentials.

Significance:

The ec2_ssrf module is a testament to the risks of SSRF within AWS environments. By guiding users through the process of exploiting such vulnerabilities, it underscores the necessity of safeguarding against SSRF. It's a potent tool, perfect for both vulnerability demonstrations and educational purposes.

The AWSNeo Framework, with its various modules, emerges as a comprehensive suite for AWS security testing. By embracing the spirit of both reconnaissance and active exploitation, it's poised to offer invaluable insights into the security posture of AWS deployments.

6 Evaluation

6.1 Experiment 1: Assessment of IAM Reconnaissance Modules

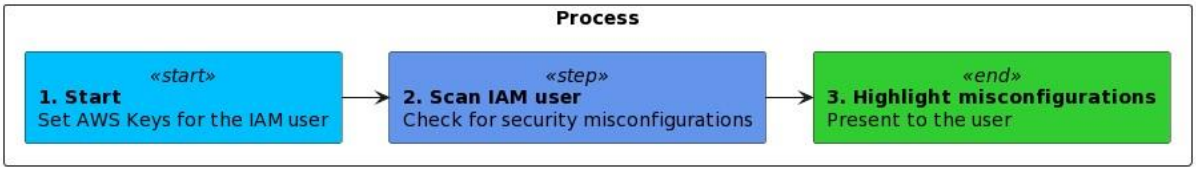

Figure 2: IAM Reconnaissance Flow

- **Tool Type:**
  In this evaluation, we employed the AWSNeo IAM recon module, juxtaposing it against the weirdALL tool, a Python-based utility for IAM reconnaissance. Although AWS CLI commands are another option, they require a manual, multi-step process. For the purposes of this test, we supplied the AWS Secret key for user "Lambda-1" to both tools. The evaluations were conducted using Kali Linux.
- **Automation Level:**
  AWSNeo effectively identifies security misconfigurations within the IAM services. Upon detecting a high-privilege vulnerability in IAM access, it prompts users to inspect the output file. In contrast, weirdALL provides detailed IAM services information but necessitates additional operations and data sifting to identify security misconfigurations.

- **Accuracy:**
  AWSNeo successfully identified misconfigured settings without any user intervention. On the other hand, weirdALL required several interactions to achieve the results that AWSNeo produced initially.
- **Scan Speed:**
  To measure execution time, the Linux "time" command was utilized. For AWSNeo, due to its interactive nature, a script with the "expect" tool was employed. AWSNeo completed its enumeration in 20.03 seconds, whereas weirdALL took 36.72 seconds. This indicates that AWSNeo is more efficient than weirdALL.

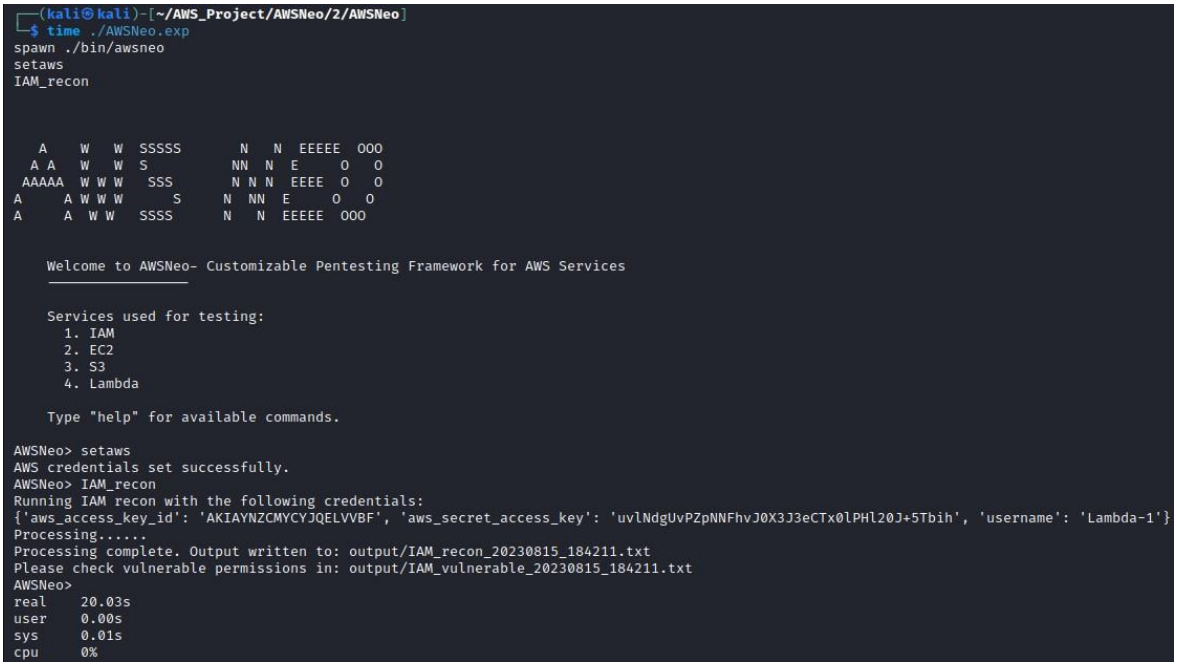

Figure 3: AWSNeo Scan Speed and Output

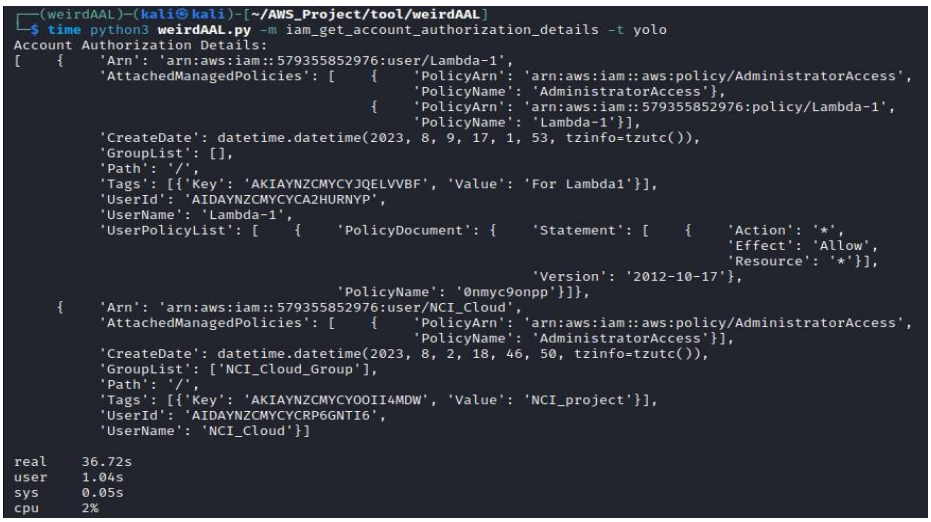

Figure 4:weirdALL Speed and Output

6.2 Experiment 2: Assessment of IAM Exploitation Modules

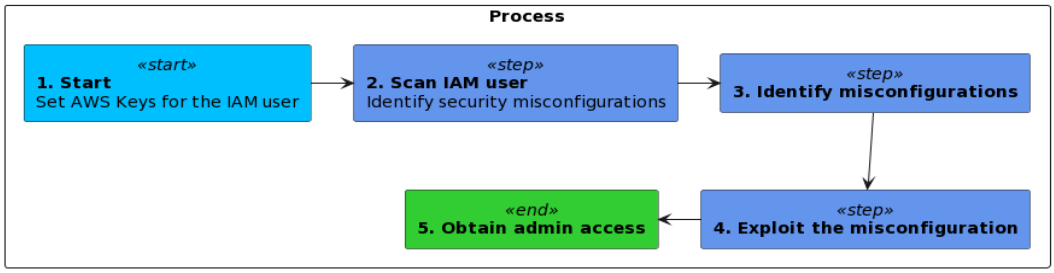

Figure 5:IAM Exploitation Flow

- **Tool Type:**
  For this test, the AWSNeo IAM exploit module was pitted against the IAMFlaw tool, a bash script utility.
- **Automation Level:**
  The AWSNeo framework streamlined the exploitation process, requiring only two steps for the IAM exploit module to run. In contrast, the IAMFlaw tool demanded several steps, rendering AWSNeo the swifter of the two.
- **Accuracy:**
  AWSNeo adeptly exploited the IAM, even with limited access. Conversely, IAMFlaw was unable to do so.
- **Exploitation Speed:**
  Utilizing the Linux "time" command to measure execution duration, AWSNeo, with its interactive interface, was paired with an "expect" tool script. AWSNeo completed the exploitation in 10.01 seconds, while IAMFlaw took 33.52 seconds, reaffirming AWSNeo's superior speed.

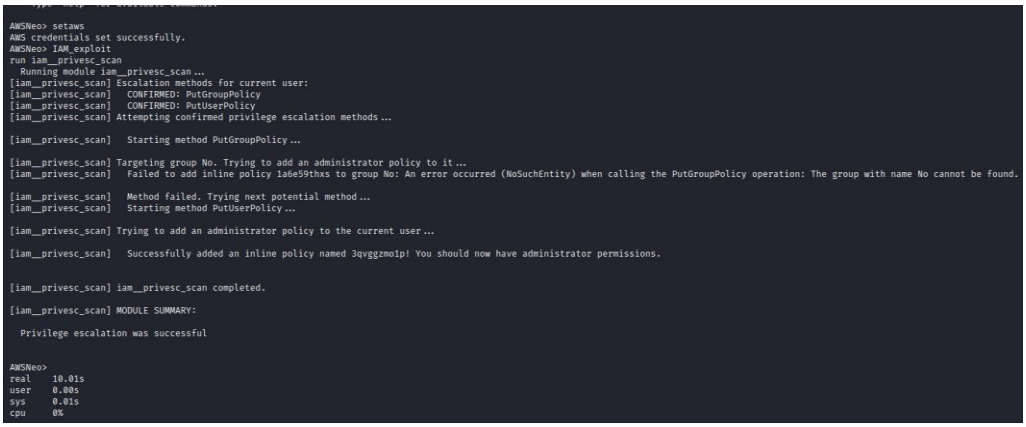

Figure 6: AWSNeo Exploitation Speed and Output

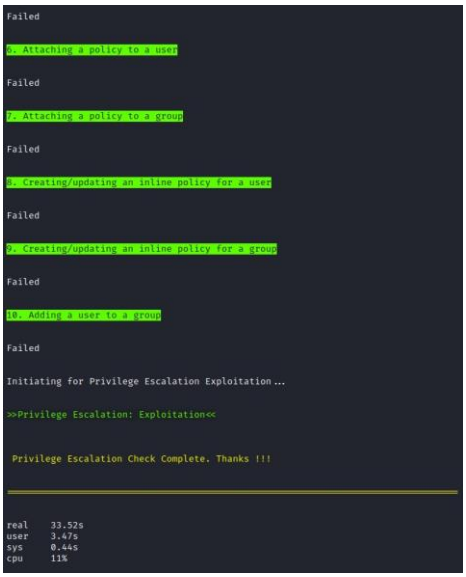

Figure 7:IAMFlaw Exploitation Speed and Output

6.3 Experiment 3: Evaluation of the EC2 SSRF Module

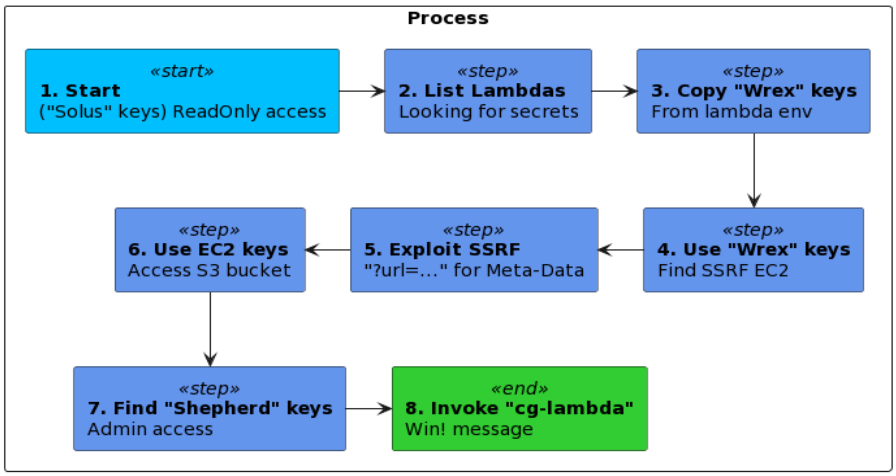

Figure 8: EC2, S3 and Lambda Exploitation Flow

- **Tool Type:**
  The AWSNeo EC2 ssrf module was the primary tool in this experiment. Comparatively, the conventional approach necessitates a suite of tools due to the complex chain of security misconfigurations. Such tools might include AWS CLI, Burp Suite, Nmap, and Peach Fuzzer, among others, which complicate and extend the process.
- **Automation Level:**
  With AWSNeo's built-in automation for such scenarios, user intervention is minimized, particularly when stacked against the conventional method's varied toolkit.
- **Accuracy:**
  AWSNeo consistently delivers precise details at each phase, streamlining the exploitation process. In comparison, the traditional approach demands cross-referencing outputs from various tools. Any misstep could mean reverting to a previous step, re-running tools, and adjusting the output.
- **Exploitation Speed:**
  AWSNeo's automation accelerates the exploitation, smoothly navigating the entire process. The traditional method, however, hinges on numerous tools with varied output formats. Extracting crucial details for subsequent tools invariably consumes more time than AWSNeo's singular, cohesive approach.

6.4 Discussions

Diving deep into the nuances of our experiments provides ample opportunities to discuss the challenges and intricacies of AWS security. The AWSNeo tool came to the forefront in terms of efficiency, especially when pitted against other tools like weirdALL and IAMFlaw. Taking a step back and critically examining our methodology, especially in the first experiment, a point of contention arises. By relying on AWS Secret keys for the evaluations, have we unintentionally introduced a potential security risk? Past experiences in the industry have repeatedly underlined the hazards of mismanaging sensitive data. While AWSNeo might be more autonomous, the manual steps inherent to AWS CLI commands, though cumbersome, do provide a direct control that many professionals might prefer. In the second experiment, the efficacy of AWSNeo was further cemented. However, there's a lingering question - in the race for automation and speed, are we potentially bypassing more subtle vulnerabilities that a hands-on tool might highlight? We can't overlook this aspect, given the varied threats looming in the digital realm. The third experiment further placed AWSNeo in the spotlight, especially in contrast with traditional methodologies that require a suite of tools. AWSNeo's one-stop-shop methodology is appealing, compared to the multi-faceted, complex approach of traditional methods.

7 Conclusion and Future Work

Reflecting upon research and through this study, our core aim was to navigate the complexities of AWS security, emphasizing tools that could optimize vulnerability assessments. AWSNeo, with its automation and speed. Yet, the focus of this research extends beyond tool superiority. It hinges upon understanding the multifaceted challenges cloud security presents in an era

where AWS has cemented its market dominance. Such prominence necessitates rigorous security mechanisms to preempt threats. However, like all research endeavors, this research isn't without its caveats. The reliance on AWS Secret keys during our experiments stands out as a potential risk, highlighting the balance between efficiency and security.

As we move towards the future, the realm of cloud security appears brimming with untapped potential. We can envision a world where tools like AWSNeo integrate predictive algorithms, perhaps harnessing the power of machine learning, to preemptively counter threats. Another avenue worth exploring is the amalgamation of AWSNeo's automation with the in-depth analysis provided by manual tools. This hybrid approach could ensure both speed and comprehensive security insights. Moreover, given the unique challenges every enterprise faces, future research could tailor solutions specifically for small and medium-sized businesses adapting to tools like AWSNeo.

In the wide scope of cloud security research, our study is but a single thread. Yet, it underscores a collective endeavor: to continuously innovate, ensuring robust cyber defenses for Cloud environments.

References

Abirami, R. *et al.* (2020) 'Detecting Security Vulnerabilities in Website Using Python'. *Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020*, pp. 844–846.

Blancaflor, E. *et al.* (2023) 'The Use of Cloud Computing and Its Security Risks in a Philippine Education System: A Literature Review'. *2023 11th International Conference on Information and Education Technology, ICIET 2023*, pp. 66–70.

Casola, V. *et al.* (2018) 'Towards Automated Penetration Testing for Cloud Applications'. *Proceedings - 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2018*, pp. 30–35.

Chang, V. and Ramachandran, M. (2016) 'Towards Achieving Data Security with the Cloud Computing Adoption Framework'. *IEEE Transactions on Services Computing*, 9(1), pp. 138–151.

Chaudhari, A.R., Gohil, B.N. and Rao, U.P. (2022) 'A Review on Cloud Security Issues and Solutions'. *Journal of Computer Security*, pp. 1–27.

Gaur, Kashish. *et al.* (2023) 'Prevention of Security Attacks in Cloud Computing'. *2023 6th International Conference on Information Systems and Computer Networks, ISCON 2023*. DOI: 10.1109/ISCON57294.2023.10112020.

Hakani, D. and Mann, P.S. (2022) 'A Comprehensive Survey on Cloud Security Mechanisms'. *International Conference on Automation, Computing and Renewable Systems, ICACRS 2022 - Proceedings*, pp. 471–475

Hu, Y., Wang, W. and Tiwari, M., 2023. Greybox Penetration Testing on Cloud Access Control with IAM Modeling and Deep Reinforcement Learning. *arXiv preprint arXiv:2304.14540*

Jansi Sophia Mary, C., Mahalakshmi, K. and Senthilkumar, B. (2023) 'Deep Dive On Various Security Challenges, Threats And Attacks Over The Cloud Security'. *2023 9th International Conference on Advanced Computing and Communication Systems, ICACCS 2023*, pp. 2089–2094.

Ksiezopolski, B. *et al.* (2022) 'Teaching a Hands-On CTF-Based Web Application Security Course'. *Electronics (Switzerland)*, 11(21). DOI: 10.3390/ELECTRONICS11213517.
Kumar, R. and Goyal, R. (2019) 'On Cloud Security Requirements, Threats, Vulnerabilities and Countermeasures: A Survey'. *Computer Science Review*, 33, pp. 1–48.

Latha, D.P.P., Pushparaj, D.J. and Helina, S.T. (2022) 'An Efficient Approach of Security Risk Mitigation in Cloud'. *Journal of Pharmaceutical Negative Results*, 13, pp. 338–342.

Mallisetty, S.B. *et al.* (2023) 'A Review on Cloud Security and Its Challenges'. *IDCIoT 2023 - International Conference on Intelligent Data Communication Technologies and Internet of Things, Proceedings*, pp. 798–804.

Mansour, I.E.A., Bouchachia, H. and Cooper, K. (2017) 'Exploring Live Cloud Migration on Amazon EC2'. *Proceedings - 2017 IEEE 5th International Conference on Future Internet of Things and Cloud, FiCloud 2017*, 2017-January, pp. 366–371.

Mohan, A., Swaminathan, G.A. and Shafana, N.J. (2022) 'Automated Tools and Techniques in Vulnerability Assessment'. *Proceedings - 4th International Conference on Smart Systems and Inventive Technology, ICSSIT 2022*, pp. 533–540.

Muralidharan, M., Babu, K.B. and Sujatha, G. (2023) 'Pyciuti: A Python Based Customizable and Flexible Cybersecurity Utility Tool for Penetration Testing'. *International Conference on Innovative Data Communication Technologies and Application, ICIDCA 2023 - Proceedings*, pp. 679–683.

Paul, P. and Aithal, P.S., 2019. Cloud security: An overview and current trend. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *3*(2), pp.53-58

Raceanu, D.N. and Marian, C.V. (2023) 'Cybersecurity Virtual Labs for Pentesting Education'. *13th International Symposium on Advanced Topics in Electrical Engineering, ATEE 2023*.

Raj, P., Shankar Kumar, R. and Kumar, A. (2022) 'Cloud Security'. *Proceedings - 2022 4th International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2022*, pp. 1890–1894

Raj, S. and Walia, N.K. (2020) 'A Study on Metasploit Framework: A Pen-Testing Tool'. *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, pp. 296–302.

Reddy, M.V. *et al.* (2023) 'A Systematic Approach towards Security Concerns in Cloud'. *Proceedings of the 2023 2nd International Conference on Electronics and Renewable Systems, ICEARS 2023*, pp. 838–843.

Rohith, D.S. *et al.* (2023) 'Review on Cloud Security Attacks and Preventions'. *Proceedings of the 2023 2nd International Conference on Electronics and Renewable Systems, ICEARS 2023*.

Roy, A. and Patil, K. (2023) 'Framework for Cloud Security Initiatives in Small and Medium-Sized Enterprises'. pp. 444–449.

Singh, S. and Kumar, D. (2023) 'Vulnerability of Cyber Security in Cloud Computing Environment'. *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 572–580. DOI: 10.1109/ICESC57686.2023.10193087.
Sureshkumar, V. and Baranidharan, B. (2021) 'A Study of the Cloud Security Attacks and Threats'. *Journal of Physics: Conference Series*, 1964(4), p. 042061

Tigner, M., Wimmer, H. and Rebman, C.M. (2021) 'Analysis of Kali Linux Penetration Tools: A Survey of Hacking Tools'. *International Conference on Electrical, Computer, and Energy Technologies, ICECET 2021*.

Tiwari, A., Patel, P. and Scientific, D.S.-. (2021) 'Vulnerability Assessment and Penetration Testing Approach Towards Cloud-Based Application and Related Services'. *Academia.EduA Tiwari, P Patel, D SharmaInternational Journal of Scientific Research in Science, Engineering and, 2021•academia.Edu*, pp. 395–403.

Yurtseven, I. and Bagriyanik, S. (2020) 'A Review of Penetration Testing and Vulnerability Assessment in Cloud Environment'. *2020 Turkish National Software Engineering Symposium, UYMS 2020 – Proceedings*.