Comparison of AWS and AZURE in Prototype for Automated Selected Security Controls

Mpho Erick Maimela, Joey Jansen van Vuuren and Michael Moeti
Tshwane University of Technology, South Africa

211167033@tut4life.ac.za
JansenVanVuurenJC@tut.ac.za
moetimn@tut.ac.za

Abstract. The cloud computing phenomenon has achieved global popularity, with enterprises increasingly relying on cloud services for day-to-day business operations. However, the rapid dissemination of new malicious code variants with zero-day assaults in the cloud creates confusion and broad worry because the attackers' motives often remain unknown. This paper discusses a safer computing platform or model that detects harmful or malicious code in a cloud environment and automatically selects the best security control for defence. Automated selection of the best security controls for real-time defence is crucial in cloud environments. The study utilizes pefile library in Python to extract signature bytes, N-gram algorithm for signature bytes segmentation, the C4.5 algorithm for constructing signature clusters, and a Python program to determine the best security control. The model was developed and tested using Microsoft Azure and the Amazon Web Services cloud infrastructure, with results demonstrating its effectiveness on both platforms in detecting malicious code and timely selecting an optimal security control for real-time defence.

Keywords: Cloud services, Security, N-gram, C4.5 algorithm

1. Introduction

Cloud computing is gaining global traction, with more businesses transferring their data and operations from traditional servers to cloud servers. This phenomenon refers to the on-demand distribution of information technology resources, which provides clients with access to a shared pool of computer resources at on-demand or pay-per-use rates (Subramanian & Jeyaraj, 2018)

According to (Asadi et al., 2019) cloud computing provides and facilitates client access to a shared pool of configurable computer resources such as networks, servers, applications, and services. Despite the benefits, security control is still a big challenge in Cloud Computing platforms (Rios et al., 2019) The continual creation of new harmful code offers a significant challenge to robust cybersecurity. Organisations must carefully select appropriate security policies to secure sensitive data and reduce potential threats. The typical procedure of determining appropriate security controls based on system technology, known vulnerabilities, and attack patterns is time-consuming, error-prone, and requires extensive knowledge (An et al., 2019). Traditional techniques for establishing security controls frequently result in considerable compromises in core functional processes and security needs (Ehrlich et al., 2019).

The continuous development of new malicious codes also puts intense pressure on software and anti-virus companies to update their database definitions from time to time, which causes a challenge to keep up with the pace at which the scripts are developed and distributed (T. Liu et al., 2022) This paper puts forward an enhanced model to detect, classify, and automatically select optimal security control to respond to malicious code attacks in cloud computing environments without human supervision. The automated security control selection is essentially to keep up with the pace of malicious code development and the inevitable changing of the industrial internet to avoid the static procedures and physical human effort to secure the dynamic applications and cloud computing environments (Gergeleit et al., 2020).

2. Background

With the rapid expansion of the industrial internet and the migration of data from traditional servers to cloud servers, cybersecurity attacks based on malicious code are becoming more sophisticated and ubiquitous than ever before. Attacks utilising unique malicious programs result in a wide range of harms, including production losses, data integrity, intellectual property theft, and reputational damage. The emergence of new malware strains complicates an already difficult task of countering attacks. Cybercriminals take advantage of several vulnerabilities in applications networks and systems on cloud platforms to pose an attack on users and services (Simou et al., 2022).

While there are various methods for analysing cloud security and combating potential assaults, existing models clearly fall short of covering all possible risks (An et al., 2019). Detecting and defending cloud computing systems against malicious code attacks has always been a challenge for security experts and businesses. Masked malicious code is the most difficult danger to detect since it masquerades as a legitimate portion of the software.

Malicious code including viruses, worms, Trojan horses, logic bombs, spyware, and rootkits, refers to programs designed and developed to purposefully cause unexpected and unpleasant events on the operating system, network, applications, or their dependencies. These attackers target flaws in systems and applications and execute malicious code without the user's consent and knowledge. (Anderson, 2010). Malware detection is a method of identifying and mitigating potential security threats by analyzing the content of a program through malware analysis, feature extraction, and classification (Aslan et al., 2023)

3. Related Work

Public clouds are shared by nature, with applications, including malicious ones, sharing physical machines and resources (Shringarputale et al., 2020). Cloud providers like AWS, Microsoft Azure, and IBM Cloud implement a shared responsibility model for cloud security (Hong et al., 2018). Today's increasingly virtualized computer systems, driven by cloud platforms like AWS and Microsoft Azure, present an interesting challenge for malware, through employing Virtual Machine (VM) detection techniques, likely to free potential malware victims (Kemkes, 2019).

A study conducted on AWS and Azure shows novice IoT programmers overlook security, highlighting the use of proper tools, settings, and standards (Corno et al., 2022). SQL injection is among common malicious code attacks, where attackers insert code into standard SQL to gain unauthorized access (Bhadauria et al., 2012). Attackers may install malicious code, potentially enabling future unauthorized access (Doan et al., 2018). Approximately 1 million malicious code variants are propagated into cloud platforms daily, with each variant and its associated family created for specific malicious intentions (Aslan et al., 2021). Herded malicious codes facilitate attacks like eavesdropping, data theft, and compromising sensitive information (Sharma & Semwal, 2022). Since adversaries inject the malicious code themselves, they are knowledgeable of the procedures to launch successful attacks (G. Liu et al., 2022).

AWS dominates 34% of the $200 billion cloud market, emphasizing the need for better intrusion detection in network security (Balajee & Kannan, 2023). In distinct studies, intrusion detection on Microsoft Azure and AWS were performed, with Microsoft Azure showing a 99.95% accuracy (Hafsa & Jemili, 2018), and intrusion detection on AWS cloud through a hybrid deep learning algorithm achieving 95% (Balajee & Kannan, 2023). AWS and Microsoft Azure offer robust administrative and security tools (Ogbole et al., 2021), however the tools are not immune to attacks. It's critical to understand threats specific to AWS, Azure, or Google Cloud Platform (George & Sagayarajan, 2023).

Boneder (2023) emphasised that future research should investigate ransomware detection on other cloud providers, like AWS and Azure, to evaluate the adaptability and effectiveness of machine learning-based methods in diverse cloud environments. This study expands detection scope and cover other malicious code threats in AWS and Microsoft Azure cloud platforms. Evaluating co-residency attacks on containers using real-world workloads demonstrated 90% success rate on both AWS and Azure (Shringarputale et al., 2020).

Yang, Zhao, and Liu (2017) developed a model to segment and classify malicious code detection using C4.5 and N-gram algorithms, and results showed improved detection accuracy, an approach implemented in this study. This study puts forward a prototype for comparative analysis on detection in both AWS and MS Azure, using C4.5 and N-gram algorithms for enhanced detection and autonomous security control selection. In a study conducted on the security solutions in AWS and Microsoft Azure, it proposes establishing patterns and mapping them to security controls (Rath et al., 2019). This study revolutionizes mapping of security controls through autonomous selection. AWS and Azure comparison across top security scanning tools revealed that each tool has its strengths and capabilities (Singh & Aggarwal, 2022), underscoring this study's approach to automate security control selection for robust cloud security posture.

4. Methodology

This study employed the Design Science Methodology (DSM). The DSM was chosen for this study because (Baskerville & Pries-Heje, 2019) indicates that design science research aims to be prescriptive, purposeful, relevant, and add value, which is an objective of this study. DSM is also most commonly used in studies aimed at designing, developing, and evaluating artefacts for practical solutions (Bisandu, 2016). This methodology

included a literature review and usability testing, and this combination allowed for improved insights into the research area of malicious code detection, as well as the development and evaluation of a prototype rather than simply understanding the phenomenon.

The DSM methodology includes an iterative process and makes provision for the inclusion of other methods e.g. machine learning techniques for the development of the artifact. Figure 1 is the architectural view of how Design Science, Literature Review and Usability testing methodologies were used that were influenced by the mixed methods approach by (Jansen van Vuuren et al., 2016)

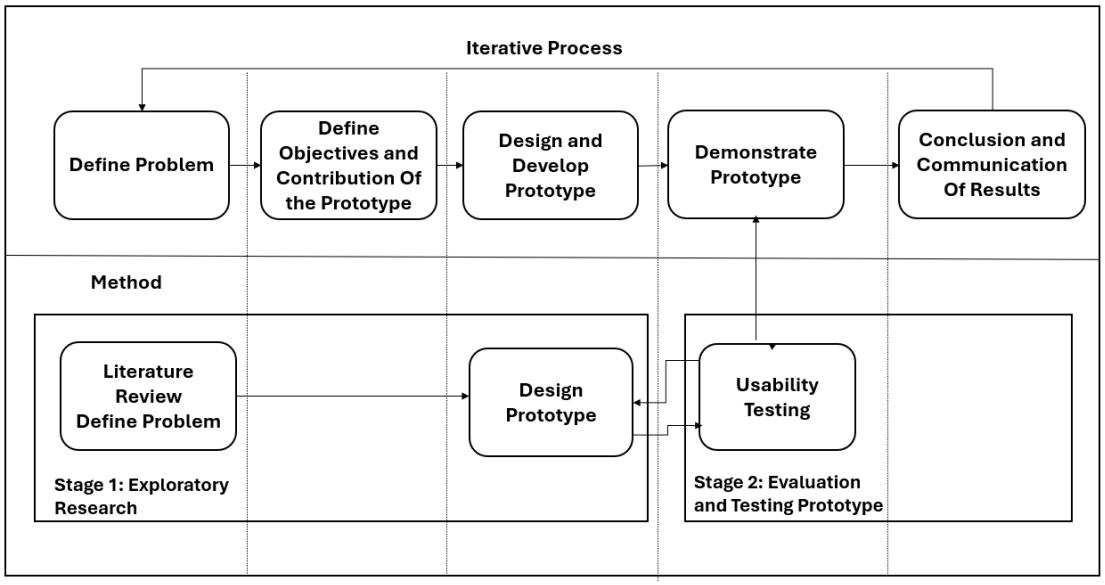

Figure 1: Design Science, Literature Review and Usability Testing

The following steps provide a quick description of how the methodology was utilised to obtain the intended results:

- Step 1: Define Problem - During this stage, we found a gap in the literature via a Literature Review, indicating the necessity to automate the detection and selection of security controls.
- Step 2: Define the objectives and contributions of the model and solution. - In this step, we defined the study's aims and analysed the relationships that were needed to create the model.
- Step 3: Design and develop the artefact - A prototype was created to demonstrate the solution.

5. The Automating Selection of Security Controls System (ASSCS) Model

To effectively detect and cluster malware families, the model employs two algorithms, namely the N-gram and C4.5 algorithms. Options were investigated and the combination of the two algorithms yielded the best performance results.

5.1 N-gram

The N-gram method, also known as the N meta-model, is an instrumental technique in the domain of Natural Language Processing (NPL) that is well-recognised for its capacity to partition strings based on their length N (Wang et al., 2020) The N-gram approach described in this research was utilised to segment signatures from a byte sequence of a malicious code script used to attack a cloud computing system. The collected signature bytes from the n-gram are sent into the classification algorithm (C4.5 algorithm), which produces a signature tree.

5.2 C4.5 Algorithm

The C4.5 algorithm is used as a Decision Tree Classifier of the signature bytes generated and gives unmatched classification from simple to complex extractions. The algorithm helped improve accuracy and ensured that malicious codes were accurately detected, as it can work with all textual data. C4.5 was mostly preferred for this study due to its capability of handling incomplete data, which reduce chances of false negatives. According to (Rahim et al., 2018), C4.5 algorithm is very impressive in classification and prediction.

5.3 Creation of the Detection Model

The model is based on three techniques, namely, the pefile Python library, N-gram, and C4.5 algorithm. The first technique, the pefile Python library, extracts signature bytes from executable files. The second technique, N-gram, segments the extracted signatures into patterns. The third technique, C4.5 algorithm, classify the patterns to determine if the code is malicious or benign. Benign code is allowed to execute, while malicious code is identified as harmful, and a Python module will be triggered to select optimal security control for the defence.

To identify and isolate the main characteristics of a virus from signature is a complex and lengthy process that involves breaking down and analyzing the signature features (Mercaldo, 2021). The model can swiftly identify new malicious code variants by leveraging N-gram's segmentation capabilities. According to (Abiola & Marhusin, 2018) understanding the signature byte, it's helpful to divide it into several patterns. The N-gram algorithm was employed to create malicious code signature patterns to boost detection of new malicious code variants by breaking down signature bytes into sequences of n-items. This approach successfully uncovered hidden virus patterns by analyzing the smallest details of the signatures, effectively identifying and mapping malicious signatures to known malicious code families. The n-items were fed into C4.5 algorithm for classification. The classified n-items were compared against the data in the malware repository.

6. ASSCS Prototype

6.1 System Architecture

The architecture in Figure 1 represents a complete system from the initial stage where a hacker launches an attack to a point where the attack is defeated, and a history of the malicious code attack is stored in the repository for future reference. The malicious code herder initiates an attack by sending a malicious code to the cloud server. The ASSCS autonomously scans and responds to suspected and foreign code and initiates a further step to determine if the code is malicious or benign. Benign code is considered as the legitimate, whereas malicious code is identified as harmful. If the code is identified as harmful, ASSCS triggers a response from the Security Controls database and sends an optimal security control for response. The defeated malicious code data is recorded in the malicious code repository for future reference.

The UML diagram shown in Figure 3 illustrates a structured approach for various stages involved in identifying, classifying, and combating malicious code. The processes are executed sequentially to address security threats posed by malicious code.

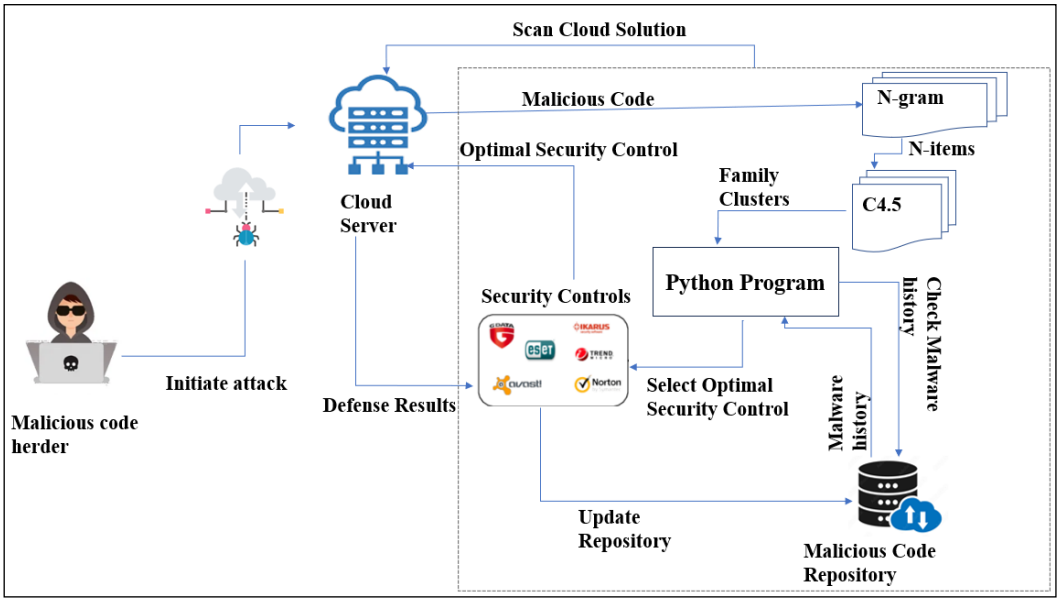

Figure 2: Proposed System Architecture: Automated Cloud Security Solution (ASSCS)

6.2 ASSCS System UML Diagram

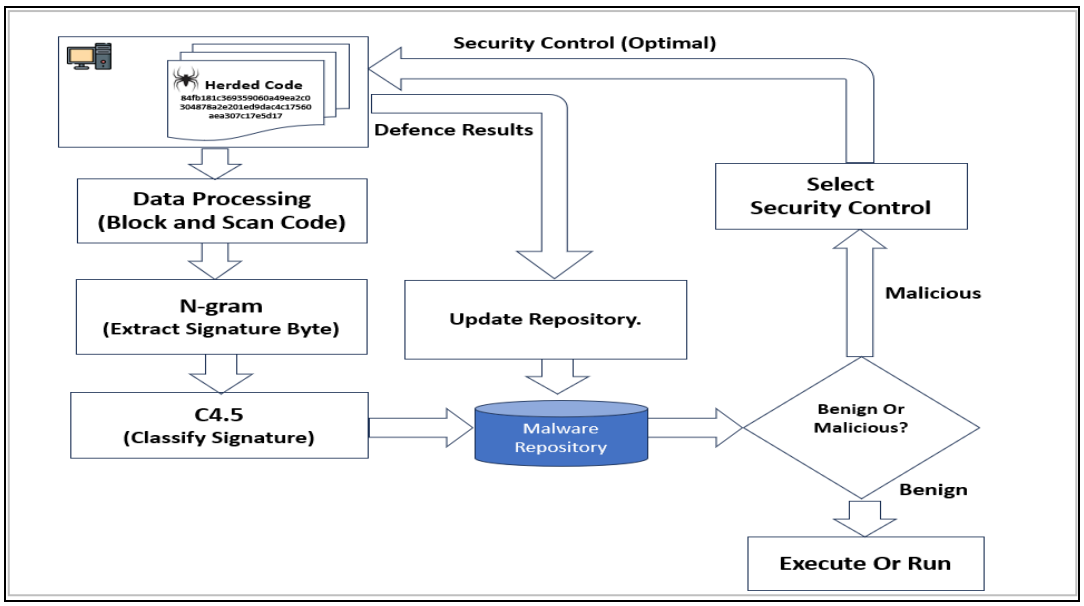

Figure 3: Stages of the developed model

The program was developed using Python programming language. Python stands out as the top choice for scientific computation, data science, and machine learning because of its high level of performance, productivity, libraries, and high-level Application Programming Interfaces (APIs) (Raschka et al., 2020)[14]. The program was developed to use the signature classified results to compare the clusters against the virus history stored in the virus repository. Newly identified segments matching known virus patterns were treated as new variants and quarantined to suppress potential harm.

The program's ability to efficiently select optimal security control is supported by a rich malware family history stored in a virus repository, which is continually auto-updated with definitions as new viruses and variants are identified. When a malicious script is detected and suppressed, detailed records of the code, including signature byte segments and the specific security control that successfully countered the threat, are stored in the repository. This historical data allows the system to recognise and effectively respond to future attacks from the same virus family or its variants by automatically selecting the security control that previously proved to be the best in dealing with the virus family.

7. Experimental Results

The system was evaluated using a dataset of 1000 codes. During the process, codes were extracted, and harmful and benign codes were separated. The evaluation recorded the malware families and the optimal security controls applied. This confirmed the system's ability to select appropriate security controls and respond to malware attacks autonomously. WEKA Explorer was used to visualise the experimental outcomes, which included loading the dataset and assessing it using the C4.5 algorithm. The visualisations differentiated between benign and harmful code, emphasising the difference between lawful and malicious code. The prototype was then tested in controlled environments such as Azure and AWS Cloud, allowing the suggested system to run without internal or external disturbance. The datasets were retrieved from GitHub and other sources.

7.1 Comparison of Results: Microsoft Azure Vs Amazon Web Services

The prototype performance is compared across two platforms based on three key aspects: classification performance, segmentation performance, and resource utilisation. Classification performance assesses the C4.5 algorithm's effectiveness on each platform using key metrics such as accuracy, precision, recall, and F1 score. Segmentation performance evaluates the N-gram algorithm's effectiveness on various n-sizes across the platforms. Resource utilisation examines the usage of resources, specifically Central Processing Unit (CPU) and Random Access Memory (RAM), within the two cloud environments.

7.1.1 Signature extractions

Portable Executable (PE) files are a major vector for advanced and prevalent malicious code attacks, which pose a significant cybersecurity threat in the cloud (Connors & Sarkar, 2023). The prototype utilizes a pefile library in Python to extract signatures designed to parse PE files. Leveraging on pefile, the prototype extract signature bytes directly from the code within the executable. The extracted signature byte is then fed into the N-gram algorithm for segmentation into smaller patterns for malware analysis.

7.1.2 Segmentation of signature byte

The N-gram algorithm was used to perform signature byte segmentation. The algorithm performance was the same in both the Microsoft Azure and AWS Cloud Platforms see Figure 4. The two cloud platforms were compared based on the ASSCS's ability to extract signature bytes and segment them into n-items.

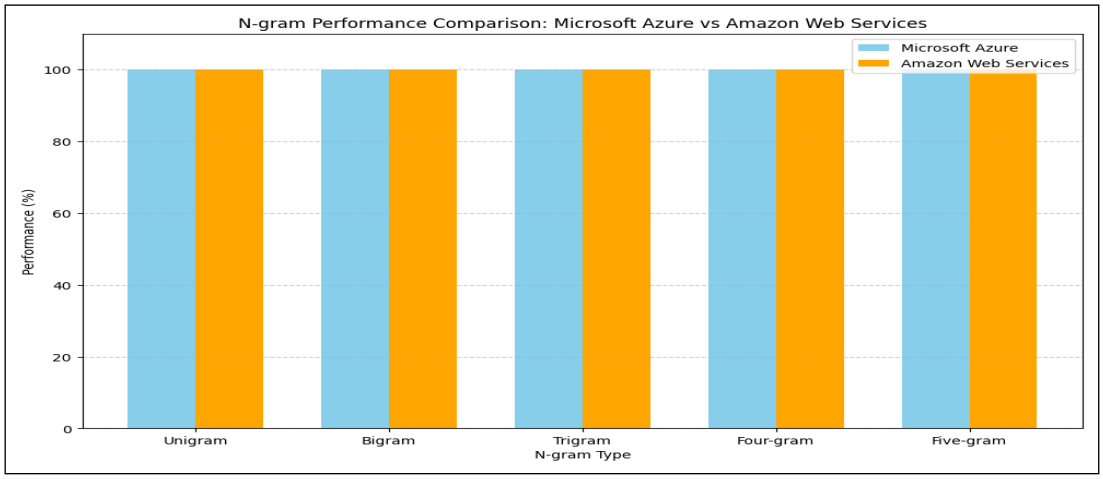

Figure 4: N-gram Performance Comparison: Microsoft Azure vs Amazon Web

From Figure 4 the performance was the same, underscoring the prototype's ability to adapt across the two cloud platforms without operational issues.

7.1.3 Classification: Microsoft Azure and Amazon Web Services using C4.5

To assess the ASSCS's adaptability and classification performance, the system was piloted on both Microsoft Azure and AWS. The performance was measured across various key metrics: precision, recall, accuracy and f1 score. Figure 6 and Figure 7 summarize the results of the evaluation done on the 2 platforms.

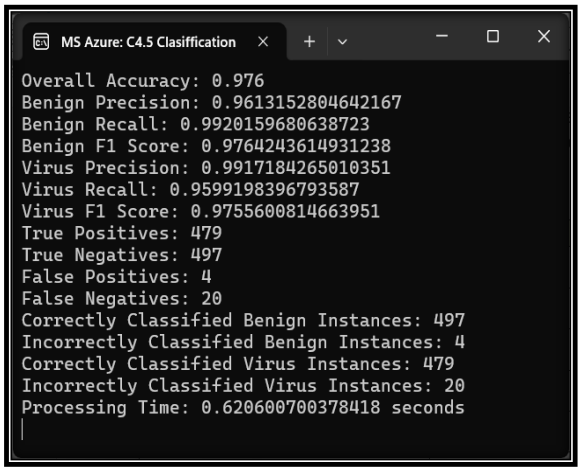

Figure 5: Microsoft Azure: C4.5 Classification

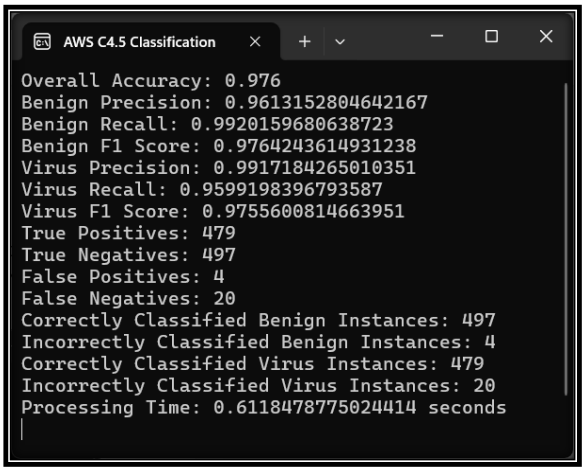

Figure 6: AWS C4.5 Classification

Below are the results compared per category in graphical representations:

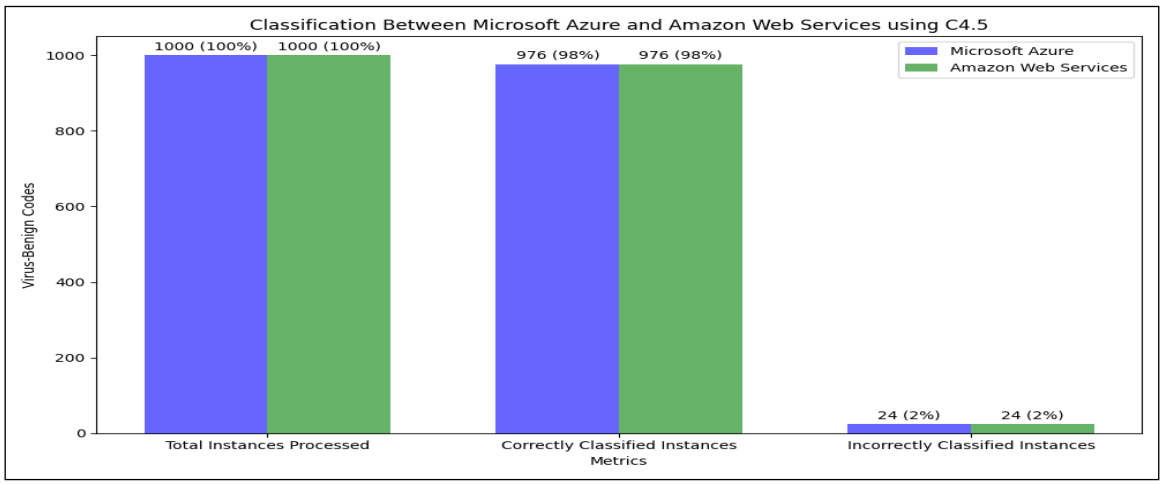

Figure 7: Instances Processed in Microsoft Azure and Amazon Web Services using C4.5

Figure 5 presents a comparison of the performance of the C4.5 algorithm based on the correct classification of the instances. Both Microsoft Azure and AWS processed a total of 1000 instances, accounting to 100% of the data. The platforms achieved a high accuracy, with 976 instances correctly classified as either benign or malicious, representing 98% for each platform. Figure 6 and Figure 7 summarize the outcome of the C4.5 comparison on both platforms.

The instances that were incorrectly classified are 24, which accounts to 2% of the total instances processed. Overall, C4.5 displayed consistency across the platforms with 98% accuracy and only 2% error rate. The results show that the C4.5 performed significantly well on both platforms, underscoring the model's capability to adapt to different cloud platforms.

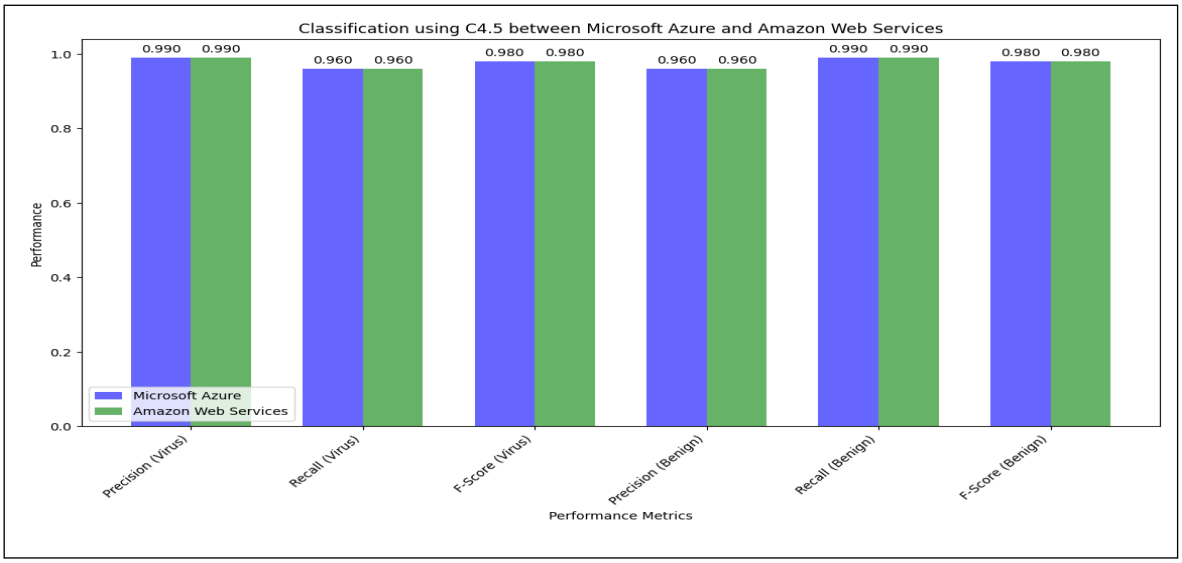

Figure 8: Classification Comparison Between Microsoft Azure and Amazon Web Services using C4.5

Figure 8 presents a comparison of the performance of C4.5 algorithm based on the key metrics: precision, recall and F1 Score for both the malicious code and benign codes.

Below is a brief explanation of the measures on virus instances precision, recall and f1-score:

- **Precision (Virus):** Microsoft Azure and AWS achieved 0.990 precision, indicating that 90% of the instances identified as viruses were truly viruses.
- **Recall (Virus):** Both the platforms achieved a recall of 0.960, meaning that they successfully identified 96% of the actual virus instances.

- **F1-Score (Virus):** The F-Score for virus classification is 98% on both platforms, which is a balanced measure of precision and recall, demonstrating high accuracy in virus code detection.

Below is a brief explanation of the measures on benign instances precision, recall and f1-score:

- **Precision (Benign):** Microsoft Azure and AWS achieved 0.960 precision, indicating that 96% of the instances identified as benign were indeed benign codes.
- **Recall (Benign):** Both the platforms achieved a recall of 0.990, meaning that both successfully identified 99% of all actual benign instances.
- **F1-Score (Benign):** The F-Score for benign classification is 98% on both platforms, which is a balanced measure of precision and recall, demonstrating high accuracy in benign code detection as well.

The results show that C4.5 performed consistently and significantly well in both the cloud platforms, with high precision, recall and F1-score for both virus and benign instances classification. The results suggest that the model is highly accurate, reliable and adaptable on both platforms.

7.1.4 Resources utilization comparison in cloud platforms

Resource utilisation is critical for measuring program performance and usability. During the experiment, the CPU and RAM performance were measured. There is a slight difference but overall, both Microsoft Azure and Amazon Web Services' performance ranges within the same measurements.

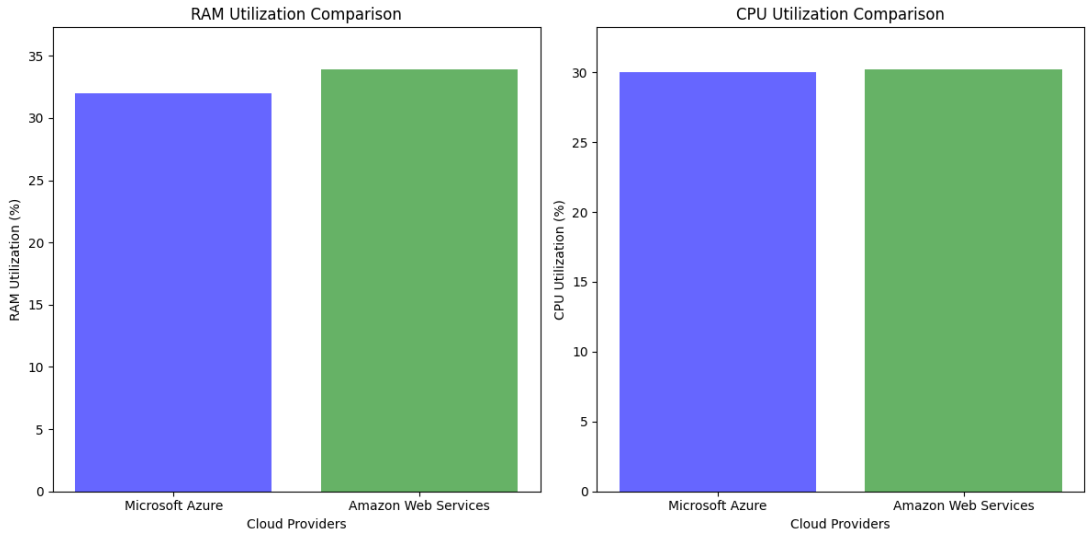

Figure 9: Resources Utilization Comparison in Cloud Platforms

7.1.5 Average processing time comparison

The time required to classify the malicious code was measured to assess the classification processing time. Most methods prioritize accuracy over speed, whereas real-time malware detection is a critical factor (Belaoued & Mazouzi, 2016). The measurements made on the two platforms show a little difference in milliseconds. Microsoft Azure outperformed Amazon Web Services. When the measurement time is converted to seconds, the processing time remains the same on both systems. Overall, the classification performance of the two platforms is good and adequate. In a real-world scenario, this will be an accurate performance evaluation, particularly given the dataset's quantity and the malicious code's intricacy. Figure 10 shows the difference between the platforms.

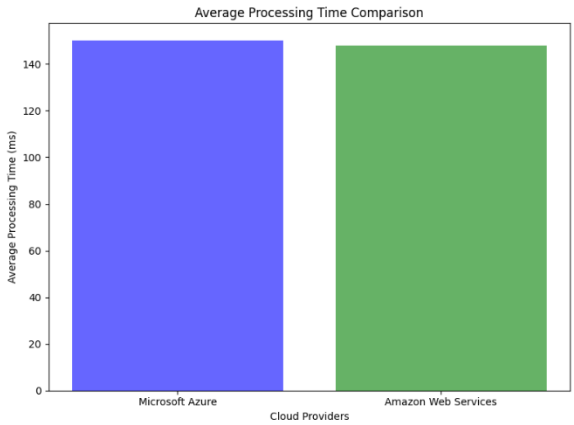

Figure 10: Average Processing Time Comparison

7.1.6 Security control selection

The prototype's ability to select optimal security control for response was evaluated using different code that mimic various families of malware. The results, as shown in Figure 11 indicates a high accuracy and precision in detecting malicious code, with similar results between the two platforms. Classification into families and triggering the security control for a response was also good. Consistency was highly maintained, and the selection of security controls was the same across both platforms.

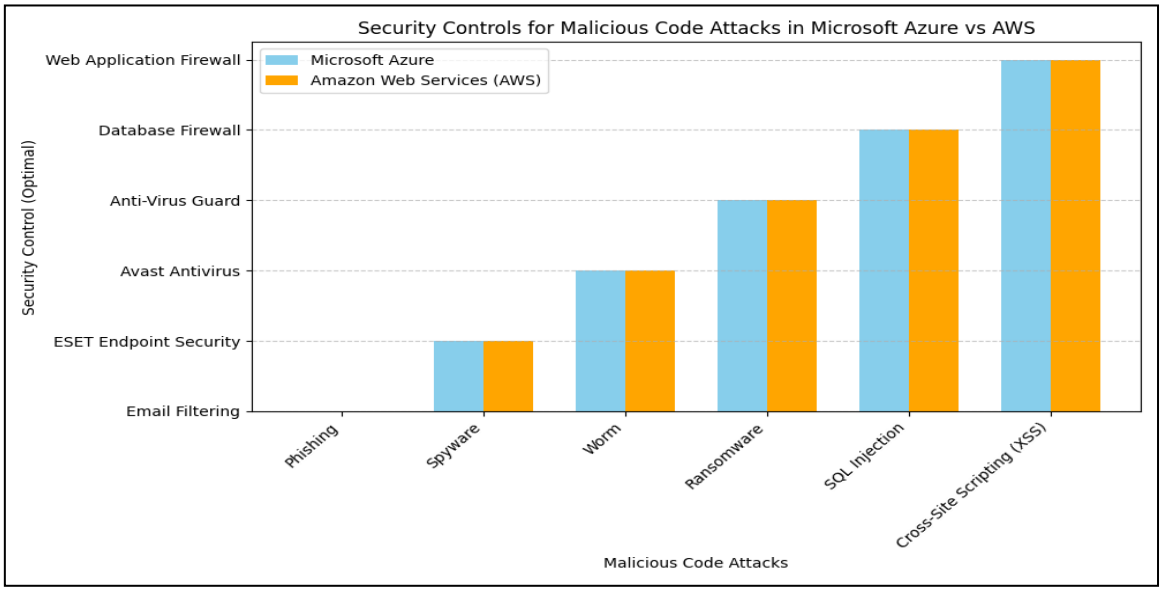

Figure 11:Malicious Code Attacks in Microsoft Azure vs AWS

7.2 Comparison Summary

Table. -1 summarises the overall performance of the classification across various metrics on both Microsoft Azure and Amazon Web Services. The performance is significantly good and commendable.

Table 1: C4.5 Classification Comparison Summary

Metric	Microsoft Azure	Amazon Web Services
Total Instances Processed	1,000	1,000
Correctly Classified Instances	976 (98%)	976 (98%)
Incorrectly Classified Instances	24 (2%)	24 (2%)
Precision (Virus)	0.99	0.99
Recall (Virus)	0.96	0.96

Metric	Microsoft Azure	Amazon Web Services
F- Score (Virus)	0.98	0.98
Precision (Benign)	0.96	0.96
Recall (Benign)	0.99	0.99
F-Score (Benign)	0.98	0.98
Average Processing Time (ms)	620.601	611.85
Resource Utilisation	RAM 32%, CPU 30%	RAM 33.9%, CPU 30.2%

Below is a brief interpretation of key metrics in the table above.

- **Precision:** Both the Cloud platforms demonstrated high precision of 99% for virus and benign codes.
- **Processing Time:** AWS displayed a faster average processing time compared to Azure.
- **Resource Utilization**: AWS showed slightly better resource utilisation efficiency, likely due to platform differences.

8. Conclusion

In this research, we introduced a new strategy for detecting and countering harmful code on cloud computing systems. The method is based on three techniques: pefile library in Python, N-gram and C4.5 algorithm. The first technique uses the pefile library to extract signature bytes from executable files; the second technique uses the N-gram algorithm to segment the extracted signature bytes into smaller and manageable patterns for computation, and the third technique employs C4.5 algorithm to classify the detected malicious codes as benign or malicious code. The prototype chooses the best security control for defence based on the classification results.

A prototype was developed and tested on two different platforms, the AWS and Microsoft Azure platforms. Both of these platforms performed very well in the identification, classification and using the security controls to counter the attacks. A comparison of the two cloud computing platforms illustrates the ASSCS's robust performance across different cloud platforms without operational issues. These results underscore the potential of the artefact to be effectively deployed in diverse cloud environments to enhance automated security control selection. This usability and adaptability give the ASSCS leverage to support a wide range of cloud service providers.

References

Abiola, A. M., & Marhusin, M. F. (2018). Signature-Based Malware Detection Using Sequences of N-grams. *International Journal of Engineering & Technology*, *7*(4.5), 120-125.

An, S., Eom, T., Park, J. S., Hong, J. B., Nhlabatsi, A., Fetais, N., Khan, K. M., & Kim, D. S. (2019, Aug. 5 2019 to Aug. 8 2019). *Cloudsafe: A tool for an automated security analysis for cloud computing.* 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) Rotorua, New Zealand.

Anderson, B. (2010). USB-based virus/malicious code launch. *Seven deadliest USB attacks. Elsevier*, 65-95.

Asadi, Z., Abdekhoda, M., & Nadrian, H. (2019). Cloud computing services adoption among higher education faculties: development of a standardized questionnaire. *Education and Information Technologies*, *25*(1), 175-191. https://doi.org/10.1007/s10639-019-09932-0

Aslan, O., Aktug, S. S., Ozkan-Okay, M., Yilmaz, A. A., & Akin, E. (2023). A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions. *Electronics*, *12*(6), 1-42, Article 1333. https://doi.org/10.3390/electronics12061333

Aslan, O., Ozkan-Okay, M., & Gupta, D. (2021). Intelligent Behavior-Based Malware Detection System on Cloud Computing Environment. *IEEE Access*, *9*, 83252-83271. https://doi.org/10.1109/access.2021.3087316

Balajee, R. M., & Kannan, M. K. J. (2023). Intrusion Detection on AWS Cloud through Hybrid Deep Learning Algorithm. *Electronics*, *12*(6). https://doi.org/10.3390/electronics12061423

Baskerville, R., & Pries-Heje, J. (2019). Projectability in Design Science Research. *JOURNAL OF INFORMATION TECHNOLOGY THEORY AND APPLICATION*, *20*(1), pp. 53 – 76.

Belaoued, M., & Mazouzi, S. (2016). A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features. *Journal of Information Processing Systems*, *12*(4), 644-660. https://doi.org/10.3745/jips.03.0058

Bhadauria, R., Chaki, R., Chaki, N., & Sanyal, S. (2012). A Survey On Security Issues In Cloud Computing. *Acta Tehnica Corviniensis– Bulletin of Engineering Tome VII [2014]*, *07*. https://arxiv.org/abs/1204.0764

Bisandu, D. B. (2016). Design science research methodology in computer science and information systems. *International Journal of Information Technology*, *5*(4), 55-60.

Boneder, S. (2023). *Evaluation and comparison of the security o erings of the big three cloud service providers Amazon Web Services, Microsoft Azure and Google Cloud Platform*

Connors, C., & Sarkar, D. (2023, December). *Machine learning for detecting malware in pe files.* International Conference on Machine Learning and Applications (ICMLA), Hyatt Regency Jacksonville Riverfront, Florida, USA.

Corno, F., De Russis, L., & Mannella, L. (2022). Helping novice developers harness security issues in cloud-IoT systems. *Journal of Reliable Intelligent Environments*, *8*(3), 261-283. https://doi.org/10.1007/s40860-022-00175-4

Doan, T. T., Safavi-Naini, R., Li, S., Avizheh, S., K, M. V., & Fong, P. W. L. (2018). *Towards a Resilient Smart Home* Proceedings of the 2018 Workshop on IoT Security and Privacy, Budapest, Hungary. ACM, New York, NY, USA.

Ehrlich, M., Gergeleit, M., Simkin, K., & Trsek, H. (2019, March 18 2019 to March 21 2019). *Automated Processing of Security Requirements and Controls for a common Industrie 4.0 Use Case* 2019 International Conference on Networked Systems (NetSys), Garching b. München, Germany.

George, A. S., & Sagayarajan, S. (2023). Securing Cloud Application Infrastructure: Understanding the

Penetration Testing Challenges of IaaS, PaaS, and SaaS Environments. *Partners Universal International Research Journal (PUIRJ)*, *02*(01). https://doi.org/10.5281/zenodo.7723187

Gergeleit, M., Trsek, H., Eisert, T., & Ehrlich, M. (2020). Modeling Security Requirements and Controls for an Automated Deployment of Industrial IT Systems. *In Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu*, *2018*, 217-231.

Hafsa, M., & Jemili, F. (2018). Comparative Study between Big Data Analysis Techniques in Intrusion Detection. *Big Data and Cognitive Computing*, *3*(1), 1-13. https://doi.org/10.3390/bdcc3010001

Hong, S., Srivastava, A., Shambrook, W., & Dumitras, T. (2018). Go Serverless: Securing Cloud via Serverless Design Patterns. 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18),

Jansen van Vuuren, J. C., Leenen, L., Grobler, M., & Dawood, Z. (2016). *Mixed Methods Research for Improved Scientific Study* (M. Baran, 1965- editor. | Jones, Janice, 1959- editor., Ed.). Information Science Reference (an imprint of IGI Global).

Kemkes, P. (2019). *Evaluation of Current Virtual Machine Detection Methods* Ruhr Universitat Bochum]. 4th GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit.

Liu, G., Liu, D., Hao, S., Gao, X., Sun, K., & Wang, H. (2022). *Ready Raider One* Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA. ACM, New York, NY, USA.

Liu, T., Neware, R., Bhatt, M. W., & Shabaz, M. (2022). A study on detection and defence of malicious code under network security over biomedical devices. *The Journal of Engineering*, *2022*(11), 1041-1049. https://doi.org/10.1049/tje2.12153

Mercaldo, F. (2021). Malware Analysis II. *Formal Methods for Secure Systems, University of Pisa*.

Ogbole, M. O., Adakole, E., Ogbole, L., & Olagesin, A. (2021). Cloud Systems and Applications : A Review. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *07*(01), 142-149. https://doi.org/10.32628/cseit217131

Rahim, R., Zufria, I., Kurniasih, N., Yasin Simargolang, M., Hasibuan, A., Utami Sutiksno, D., Freedom Nanuru, R., Nikolas Anamofa, J., Saleh Ahmar, A., & Daengs Gs, A. (2018). C4.5 Classification Data Mining for Inventory Control. *International Journal of Engineering & Technology*, *7*(2.3), 68-72. https://doi.org/10.14419/ijet.v7i2.3.12618

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, *11*(4), 1-44. https://doi.org/10.3390/info11040193

Rath, A., Spasic, B., Boucart, N., & Thiran, P. (2019). Security Pattern for Cloud SaaS: From System and Data Security to Privacy Case Study in AWS and Azure. *Computers*, *8*(2), 1-28. https://doi.org/10.3390/computers8020034

Rios, E., Iturbe, E., Larrucea, X., Rak, M., Mallouli, W., Dominiak, J., Muntés, V., Matthews, P., & Gonzalez, L. (2019). Service level agreement-based GDPR compliance and security assurance in(multi)Cloud-based systems. *IET Software*, *13*(3), 213-222. https://doi.org/10.1049/iet-sen.2018.5293

Sharma, H. C., & Semwal, P. (2022). Review of Cloud Computing Data Security and Threats. *International Journal of Creative Research Thoughts*, *10*(01), 290-295.

Shringarputale, S., McDaniel, P., Butler, K., & La Porta, T. (2020). *Co-residency Attacks on Containers are Real* Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, Virtual Event, ACM, NewYork, NY, USA.

Simou, S., Kalloniatis, C., Gritzalis, S., Katos, V., & Psalidas, M. (2022). Revised forensic framework validation and cloud forensic readiness. *International Journal of Electronic Governance*, *14*(1/2). https://doi.org/10.1504/ijeg.2022.123254

Singh, A., & Aggarwal, A. (2022). A Comparative Analysis of Veracode Snyk and Checkmarx for Identifying and Mitigating Security Vulnerabilities in Microservice AWS and Azure Platforms. *An Open Access Journal*, *03*(02), 232-244.

Subramanian, N., & Jeyaraj, A. (2018). Recent security challenges in cloud computing. *Computers & Electrical Engineering*, *71*, 28-42. https://doi.org/10.1016/j.compeleceng.2018.06.006

Wang, H., He, J., Zhang, X., & Liu, S. (2020). A short text classification method based on N-gram and CNN. *Chinese Journal of Electronics*, *29*(2), 248-254.