# Network Threat Detection in IaaS Environments Based on Flow Log Data

Hubert Wójcik, Mirosław Roszkowski, and Andrzej Mycek

*Cracow University of Technology, Cracow, Poland*

https://doi.org/10.26636/jtit.2026.3.2628

**Abstract — The rapid growth of cloud services has led security monitoring to rely mainly on limited telemetry data furnished by cloud service providers. Flow logs, such as VPC Flow Logs in Amazon Web Services, are one of the key sources of information about network traffic in IaaS environments. In this study, a fully automated experimental environment was designed and deployed in the AWS cloud using the infrastructure-as-a-code approach with Terraform. The environment includes a virtual network, flow logging mechanisms, and controlled attack scenarios generating characteristic traffic patterns, such as port scanning, brute-force authentication attempts, and data exfiltration. The collected data was analyzed using cloud-native tools, in particular Amazon Athena, which enabled a detailed investigation of anomaly detection based on flow-level metrics. The results confirm that selected threats can be effectively detected using traffic metadata, including the number of unique destination ports, the repetition of communication attempts, the volume of transferred data and the temporal regularity of flows. The analysis demonstrates that flow logs alone are not sufficient to clearly distinguish between malicious activity and legitimate operations with similar characteristics, highlighting inherent limitations of telemetry in the IaaS model. The paper outlines directions for future work, including correlation with additional log sources and integration with ML models to improve detection accuracy and reduce false positives.**

***Keywords — AWS, cloud security, IaaS, intrusion detection, network traffic analysis***

## 1. Introduction

The rapid growth of cloud computing services has led to an increasing number of information systems operating based on the IaaS model. Despite their numerous advantages, such as scalability and flexibility, these environments are exposed to a wide range of security threats, particularly at the network level. Network traffic analysis is one of the key techniques for identifying malicious activities and assessing the overall security posture of an infrastructure. The complexity of modern cloud environments makes monitoring a key component in ensuring security, reliability, and compliance with organizational policies. Observability of distributed and dynamically changing cloud resources remains a key challenge [1].

In the context of cloud environments, especially within the IaaS model, network traffic analysis is critical given the highly dynamic nature of resources and the limited visibility provided by traditional network inspection mechanisms. The ephemeral lifecycle of cloud resources and the abstraction layers introduced by cloud providers restrict direct access to network-level telemetry, thereby increasing the complexity of effective security monitoring.

Cloud services can be delivered through various service models that differ in the division of responsibilities between the service provider and the end user. The most commonly adopted models include IaaS, Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

The IaaS model provides users with access to fundamental infrastructure resources, including virtual machines, virtual networks, and data storage systems. The responsibility for configuring operating systems, implementing network security controls, and securing deployed applications depends, largely, on the user. Although this model offers great flexibility, it also requires a systematic and conscious approach to security management [2].

From a security perspective, the concept of shared responsibility is paramount. According to this model, the cloud service provider is responsible for securing the underlying physical infrastructure, while the user is responsible for securing the resources deployed within the service. In the IaaS model, this responsibility includes, among other things, setting network security rules, implementing access control mechanisms, and continuously monitoring network traffic.

Network traffic in cloud environments differs significantly from that observed in traditional data centers. This distinction arises primarily from resource virtualization, dynamic infrastructure scaling, and software-level network segmentation. As a result, traffic patterns in cloud-based systems are more fluid and less predictable than those in static on-premises environments. In cloud environments, two fundamental types of network traffic can be distinguished: external traffic, which occurs between cloud-based resources and the public network, and internal traffic, which occurs within the cloud infrastructure between individual virtual resources [3].

Despite the increasing adoption of flow log–based telemetry mechanisms, relatively few research papers analyze the detection of multiple threat categories exclusively using such data sources. The existing literature focuses primarily on ML-based approaches for specific attack scenarios, such as distributed denial-of-service (DDoS) attacks or port scanning, while paying considerably less attention to the simultaneous analysis of port scanning, brute force attacks, beaconing, and data exfiltration within a unified IaaS environment.

Survey papers on IaaS security focus on general aspects of infrastructure layer security, such as cryptography and insider threats, rather than on the limitations of network telemetry and their impact on the detection of network-based threats.

Furthermore, research on anomaly detection using VPC Flow Logs has gained momentum only in recent years, predominantly in the context of Internet of Things (IoT) environments or machine learning (ML) detection techniques. However, there remains a lack of systematic analysis that addresses multiple classes of network attacks within a cohesive AWS IaaS architecture [4].

In this work, an experimental approach was adopted to evaluate the feasibility of using network flow logs – a primary source of threat intelligence in IaaS environments. In contrast to existing research, which typically focuses on individual attack types or relies on advanced ML models, the analysis in this study emphasizes the identification of network traffic patterns characteristic of different threat classes, by relying on analytical methods.

The main contribution of this work is the design of a fully controlled experimental environment in the AWS cloud and the demonstration of the extent to which anomaly detection is possible using network traffic metadata only, without access to full packet payloads.

## 2. Related Works

Research on threat detection in IaaS cloud environments increasingly focuses on network traffic objects, such as flow logs. The growing adoption of this approach is driven by technical limitations associated with full packet inspection, as well as by the scale and dynamic nature of modern cloud infrastructures. Numerous studies emphasize that flow logs provide a lightweight, scalable means of monitoring network traffic in the cloud, which differs significantly from traditional methods based on packet capture (PCAP) or NetFlow.

For example, studies conducted in IoT environments clearly demonstrate that VPC Flow Logs provide only basic metadata, such as packet and byte counts, source, destination addresses, and protocol information. Additionally, the flow logs are segmented according to time-based windows, thus hindering session reconstruction and comprehensive communication analysis. Despite these limitations, appropriate aggregation and transformation of flow logs, such as bidirectional flow reconstruction, can significantly improve the effectiveness of anomaly detection in cloud environments. Cloud service providers recommend using flow log-based monitoring for analyzing unauthorized traffic, detecting anomalies, and correlating events within security information and event management (SIEM) systems.

Despite the growing number of publications on threat detection in cloud environments, there remains a noticeable lack of studies that systematically analyze the limitations of relying solely on network flow logs for detecting multiple classes of attacks simultaneously in IaaS environments.

Existing research primarily focuses on improving detection performance through ML models. However, the interpretability of the results and the feasibility of applying simpler analytical approaches in production environments are rarely examined.

Therefore, a justified need exists for an analysis that relies on an experimental approach using real cloud infrastructure and focuses on the practical aspects of threat detection under conditions of limited data visibility.

### *2.1. Anomaly Detection Using ML Models*

The vast majority of contemporary research focuses on the automatic detection of threats using ML algorithms that analyze network metadata only. However, the literature also presents more advanced approaches that leverage network flows and native cloud observability services to design autonomous threat detection systems. Particularly noteworthy are studies that concentrate on combining system-level telemetry with network metadata and on applying ML models to enhance attack detection effectiveness.

An example of such an approach is an incident detection system based on AWS services, which integrates data from CloudWatch and VPC Flow Logs with an analytical pipeline that uses Amazon SageMaker and serverless services such as AWS Lambda and SNS. The system is designed to be fully autonomous and adaptive, addressing the limitations of traditional rule-based monitoring systems, including static CloudWatch alarms and signature-based mechanisms, which struggle to detect complex and evolving threats.

These studies demonstrate that enriching flow log analysis with ML techniques – such as isolation forest, deep autoencoders, random forest, and XGBoost – significantly improves detection effectiveness, even for attacks that are difficult to capture while relying on classical methods. This includes brute-force attempts, port scanning, data exfiltration, and DDoS attacks.

The experimental evaluation was conducted using a hybrid dataset comprising more than 2.8 million records collected during both regular operation and simulated attack scenarios, enabling a reliable assessment of model performance. Among the evaluated models, XGBoost proved to be the most effective, achieving high classification accuracy and low detection latency, while substantially reducing false positives. Feature importance analysis revealed that key indicators for anomaly detection include flow characteristics such as traffic volume, flow duration, and destination port entropy. The findings of these studies are highly relevant to the present work, as they show that despite the limited informational scope of flow logs, their integration with data processing techniques and ML models can enable effective, scalable and cost-efficient threat detection. At the same time, they highlight the growing role of automation and ML-based approaches in monitoring cloud environments, providing an important complement to the research on classical methods used for analyzing anomalous traffic patterns in IaaS infrastructures [5]. The authors of [4] presented a survey of in-

trusion detection methods based exclusively on data derived from VPC Flow Logs. The work indicates that analyzing network traffic metadata can be an effective means of identifying security incidents, even in environments where packet inspection is not feasible or is significantly constrained. Flow-based security analysis has also been investigated in forensic contexts. In [6], the concept of "superflows" is proposed, which aggregates related network flows to facilitate threat searches, incident investigations and traffic pattern analysis. The work demonstrates that higher-level aggregation of flow metadata may significantly improve the efficiency of security analysis in large-scale network environments.

Recent studies have also demonstrated the effectiveness of flow-based approaches for anomaly detection in large-scale data centers and cloud environments. Such approaches can provide scalable threat detection capabilities without requiring deep packet inspection [6], [7]. The growing scale of modern cloud infrastructures has also motivated research into large-scale anomaly detection and diagnosis mechanisms relying on network telemetry collected from production cloud environments. Such approaches enable the identification of network- and application-level anomalies across millions of cloud resources [8].

### *2.2. Security in the Infrastructure-as-a-Service Model*

Security in the IaaS model has been extensively studied by researchers in cloud computing, particularly those who focus on the infrastructure layer. In [9], an industry-driven case study is presented accompanied by a large, real telemetry dataset aimed at enabling research on anomaly detection in complex cloud environments. The authors emphasize that contemporary large-scale cloud systems (LCSs) are becoming increasingly complex and that their reliability largely depends on effective early anomaly detection. Unfortunately, the scientific literature lacks large and realistic datasets that could serve as benchmarks for evaluating anomaly detection methods.

To address this research gap, the authors of [9] released a comprehensive data set collected from IBM Cloud over a period of 4.5 months. The data set comprises 39 365 records and 117 448 features, spanning various types of telemetry data and operational metrics. This data set enables analysis of cloud behavior under both normal operating conditions and during system failures and anomalies. In their work, the authors also demonstrated the application of ML models for anomaly detection in their proprietary dataset. They discussed the data preprocessing pipeline, challenges arising from the high dimensionality and complexity of the data, and issues typical of telemetry analysis in distributed environments.

The applied ML models are intended to support system health monitoring and improve the effectiveness of early detection of abnormal system behaviors. In the context of IaaS environments, it is essential to recognize that cloud telemetry is subject to inherent limitations. Article [10] discusses the lack of complete control over the underlying cloud infrastructure, the constraints resulting from access to only partial datasets, and the influence of the cloud service provider's architecture, all of which significantly affect the ability to perform a precise and comprehensive analysis of security events.

Previous work [11] emphasized that the IaaS architecture inherently introduces significant security limitations, including those resulting from tenant isolation, limited visibility into network traffic, and dependence on the hypervisor. These factors collectively translate into substantial constraints regarding the telemetry available to cloud users.

Previous studies have shown that visibility into network communications remains one of the key challenges in native cloud environments. The growing reliance on virtualization mechanisms and software-defined networking can significantly limit the availability of security-relevant telemetry, making network traffic metadata an extremely important source of information for anomaly detection and security monitoring [12].

The challenges associated with securing cloud environments are not limited exclusively to the network layer. Recent research on managed Kubernetes services in public cloud environments [13] demonstrates that effective protection of native cloud applications requires integrating multiple security mechanisms, including identity management, compliance assessment, auditing, system hardening, and continuous monitoring. These findings further emphasize the importance of telemetry and security visibility as fundamental components of cloud security architectures.

The scientific literature also describes solutions that leverage native cloud provider security services. An example is the integration of AWS Security Hub and Amazon GuardDuty, which centralizes threat intelligence, event correlation, and the automation of security incident detection processes [14]. Such solutions leverage multiple telemetry sources available in cloud environments and serve as an important reference point for analysis that relies exclusively on AWS VPC Flow Logs data.

## 3. Environment Architecture

The objective of this research was to create a cloud-based infrastructure capable of generating, collecting and analyzing network traffic to detect selected security threats. The environment architecture was designed to support both the reproduction of realistic attack scenarios and the analysis of network traffic based on flow log data. The environment was implemented using the IaaS model and consists of a logically isolated virtual network, compute instances that serve as test resources, and components responsible for monitoring, security event detection, and data analysis. The overall architecture is illustrated in Fig. 1.

The core component has the form of a logically isolated virtual network within which all compute resources are deployed. This network constitutes an isolated address space that enables controlled observation of network traffic both within the infrastructure and in communication with the pub-

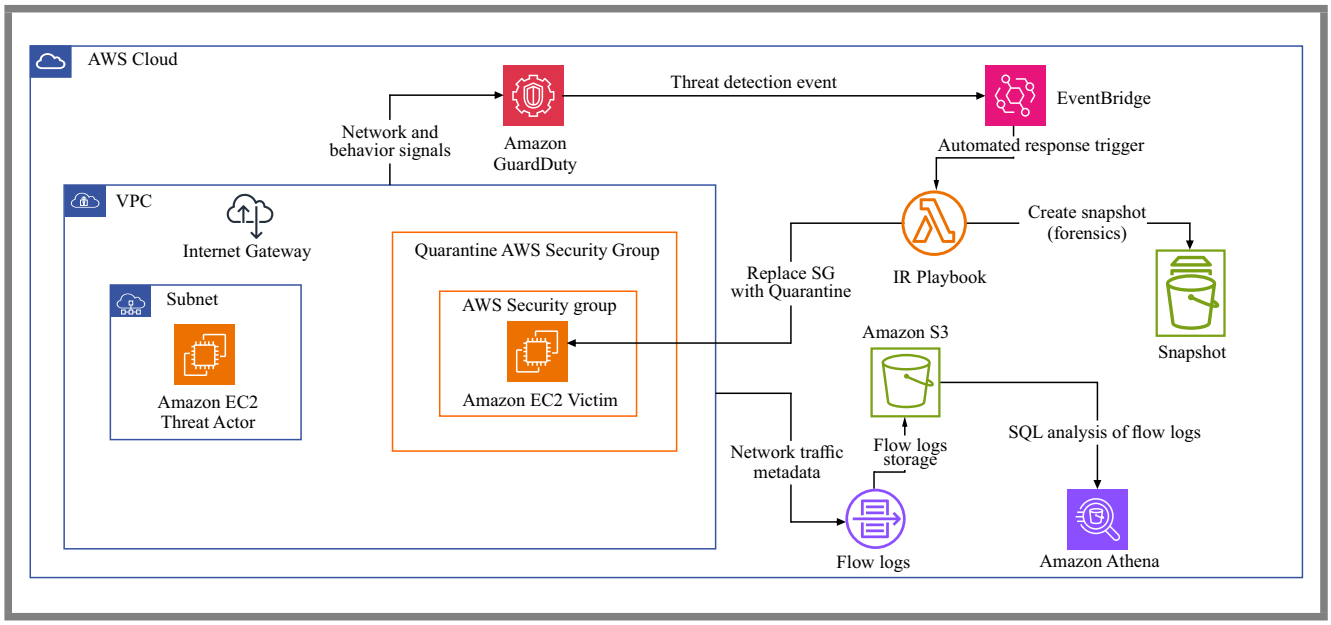


**Fig. 1.** Research environment architecture in AWS Cloud.

lic network. Within the virtual network, a subnet was created to host the compute instances. Access to the public network is provided through an Internet gateway, enabling inbound and outbound traffic. Network communication between resources, as well as communication with external networks, is governed by firewall mechanisms implemented as security groups.

Figure 1 illustrates two compute instances serving different roles. One instance generates network traffic, while the other serves as the target resource for executing the test scenarios. This separation of roles enables unambiguous analysis of both the direction and the characteristics of the observed network traffic. The experimental environment enables the reproduction of realistic threat scenarios while maintaining full control over the generated network traffic. This, in turn, enables consistent and comparable input data for further analysis, which is critical to evaluating the effectiveness of anomaly detection.

### *3.1. Network Traffic Generation, Logging Mechanism, and Threat Detection*

Network traffic generated within the research environment includes internal communication between compute instances and connections established via the public network. Every connection that traverses the network infrastructure is recorded in flow logs. The flow log metadata describe network connections, such as source and destination IP addresses, ports, protocols, packet counts, and data volumes. These data do not include packet payloads; however, they enable the identification of characteristic communication patterns.

The collected flow logs are stored in a centralized data repository, enabling further analysis using analytical tools. This approach supports both retrospective network traffic analysis and the correlation of events across different time intervals. Within the architecture of the research environment, a threat detection mechanism is implemented based on the analysis of network and behavioral signals. Events indicating potential security breaches can be generated based on observed network traffic patterns as well as other environmental parameters.

The collected network flow logs are analyzed using a tool that supports the execution of analytical queries. Analysis is performed using SQL-based queries, allowing data filtering, aggregation, and correlation across different time intervals.

This approach enables the identification of anomalies in network traffic, such as an unusually high number of connections, unexpected data volumes, or recurring communication patterns. The results of the analysis provide a basis for further investigation and evaluation of the effectiveness of network flow logs for threat detection.

## 4. Implementation

The infrastructure was deployed using an Infrastructure-as-Code (IaC) approach, which enabled repeatable resource provisioning, change version control, and reduced errors resulting from manual configuration. The implementation leveraged HashiCorp Terraform, with the configuration structured into an environment layer and a set of modular components corresponding to the individual elements. IaC makes it possible to manage the entire infrastructure (e.g., virtual machines, load balancers, virtual networks, and other services) by writing source code [15], [16].

HashiCorp Terraform is a tool that enables the declarative definition of an infrastructure in which the desired state of resources is described using configuration files (Fig. 2). Based on these definitions, the tool generates an execution plan and subsequently applies it to the target cloud environment.

```
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 5.0.0"
    }
  }
}
```

**Fig. 2.** Version and provider definition.

This approach makes it possible to recreate infrastructure in a controlled and repeatable manner. Within the project, Terraform was used to deploy key components of the environment, including the network layer, compute resources, network traffic logging mechanisms, and elements supporting incident response. This approach enabled consistent configuration management throughout the architecture and facilitated modifications during the project implementation [17]. To ensure environmental consistency and version control, explicit declarations of the required Terraform version and the cloud provider version were applied.

The environment layer serves as the entry point for infrastructure deployment and is responsible for integrating individual modules into a cohesive architecture. Within the `envs/dev` directory, the provider configuration, environment variables, and module invocations are defined that implement specific components of the research environment. The use of a separate environment layer enables straightforward adaptation of the infrastructure to different run-time contexts by modifying variable values, without the need to alter module definitions.

This approach also improves the readability of the configuration and simplifies the management of dependencies between resources. The configuration of the cloud provider and the required tool versions was defined in dedicated configuration files within the environment layer. Environment-specific parameters, such as the deployment region and network settings, were declared as input variables, allowing them to be easily modified without affecting the overall structure of the project.

### *4.1. Implementation of the Network Layer*

The network module is responsible for provisioning the foundational network layer of the experimental environment. As part of the implementation, a virtual private cloud (VPC) was defined together with two subnets: a public subnet and a private subnet. This separation enables explicit control over network traffic direction and improves resource isolation, which is essential both from a security perspective and for conducting controlled network traffic experiments.

The network layer was designed to provide Internet access for resources deployed in the public subnet while preserving the isolation of the private subnet, which has no direct connection with the Internet. This design reflects a typical architecture commonly observed in production cloud environments, where publicly accessible components are separated from internal resources.

The virtual network was created using the `awsvpc` resource with DNS support enabled, allowing proper name resolution and simplifying the instance configuration. The VPC and subnet address spaces were parameterized using variables `vpccidr, publicsubnetcidr and privatesubnetcidr` variables, allowing for easy modification of the network topology without altering the module's internal logic.

To simplify deployment, a single availability zone (AZ) was used, dynamically selected from those available within the target region. Within the module, two subnets with distinct roles were defined. The public subnet was configured to allow instances launched within it to receive public IP addresses, thereby enabling inbound and outbound Internet connectivity. The private subnet serves as an internal segment intended for resources that should not be directly exposed to the public network.

This network segmentation supports the testing of attack scenarios originating from external traffic and the observation of internal traffic flows between resources within the infrastructure. The overall design aligns with common best practices employed in cloud environments such as Amazon Web Services, while remaining sufficiently flexible for experimental and research-oriented use cases.

### *4.2. Computing Instance Definition and Log Access Controlling*

As part of the module, two EC2 instances were provisioned, each serving a distinct functional role. The attacker instance is used to generate network traffic in accordance with predefined test scenarios, while the victim instance represents the target resource against which the effects of network activities are observed. The use of role-based instance tags facilitates automation in subsequent stages of the project, particularly with respect to incident response mechanisms.

A key aspect of the configuration is the assignment of instances to appropriate subnets and security groups. The attacker instance is deployed within a public subnet, ensuring its ability to communicate with external networks. The victim instance is launched according to the module configuration, with its placement in a public or private subnet determined by the adopted architectural assumptions. In this study, the placement is aligned with the architecture diagram of the experimental environment to ensure consistency.

Access to write data to the S3 storage resource is restricted exclusively to the service responsible for delivering network flow logs. To achieve this, a dedicated resource policy was defined, allowing object-write operations only for the appropriate system service and enforcing the use of required access control attributes.

This approach ensures that log data cannot be modified or written by unauthorized entities while simultaneously satisfying the technical requirements of the network traffic logging mechanism.

### *4.3. Test Scenarios and Network Traffic Generation*

The test scenarios were designed to generate characteristic traffic patterns that can be identified using data derived from network flow logs. All scenarios were executed in a controlled experimental environment using the previously deployed computing instances assigned the roles of attacker and victim. Each scenario was conducted in a repeatable manner while maintaining constant environmental parameters, thus allowing for a reliable comparison of the results between individual tests.

To establish a baseline for analysis, network traffic generated during normal instance operation was first recorded. This reference traffic includes standard administrative communication, as well as connections initiated by the operating system. Such traffic is characterized by a low number of connections, a stable data volume, and the absence of abrupt temporal variations. The recorded baseline traffic served as a reference point for subsequent analyses of scenarios that introduce anomalous network behavior.

One of the scenarios analyzed involved port scanning of the target instance. This activity generates a large number of short-lived connections initiated within a narrow time window, often directed to multiple destination ports. In network flow logs, port scanning manifests itself as an increased number of connection attempts, low data transfer volumes, and short session durations. A distinguishing feature of this scenario is the repetition of source IP addresses combined with varying destination ports.

Another test scenario consisted of repeated authentication attempts against a network service running on the target instance.

This type of activity produces a sequence of connections to the same destination port, typically occurring at regular time intervals. In flow log data, such traffic is characterized by a high number of connections targeting a single port, moderate session durations, and recurring communication patterns.

All experimental evaluations were conducted in the controlled AWS cloud environment described in the previous sections. The analysis was based on data collected from AWS VPC Flow Logs, which were aggregated and stored in a centralized data repository. The stored flow records were subsequently queried and analyzed using Amazon Athena.

Each experimental scenario was executed under identical infrastructure conditions to ensure comparability of observed network traffic patterns. Additionally, baseline traffic reflecting the normal operation was captured to serve as a reference point. This baseline enabled the identification of the differences between legitimate network communication and anomalous traffic.

The objective of this study was not to evaluate the detection performance of a specific algorithm. Instead, it focused on identifying network traffic characteristics that may serve as reliable indicators of cyber threats using solely metadata extracted from network flow records.

```
CREATE EXTERNAL TABLE IF NOT EXISTS vpc_flow_logs (
  version INT,
  account_id STRING,
  interface_id STRING,
  srcaddr STRING,
  dstaddr STRING,
  srcport INT,
  dstport INT,
  protocol INT,
  packets BIGINT,
  bytes BIGINT,
  start_time BIGINT,
  end_time BIGINT,
  action STRING,
  log_status STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ' '
LOCATION 's3://network-analysis-dev-flowlogs-260905331134/';
```

**Fig. 3.** Definition of an external table for network flow logs.

## 5. Network Traffic Analysis

The network traffic analysis was conducted on data collected in the form of network flow logs. AWS Athena was used as the analytical tool, enabling the execution of SQL-based analytical queries without the need for prior data preprocessing. The objective of the analysis was to identify characteristic network traffic patterns generated by individual test scenarios and to evaluate the feasibility of anomaly detection based solely on traffic metadata.

### *5.1. Data Pre-processing for Analysis*

The first step of the analysis involved defining the data structure that represents the flow logs. For this purpose, an external table was created, pointing to the location of the data stored in the S3 resource (Fig. 3). The table reflects the core fields of the flow logs, including source and destination IP addresses, ports, data transfer volumes, and timestamps.

The anomaly analysis was performed by examining specific traffic characteristics for each attack scenario using AWS VPC FlowLogs. In the port scanning scenario, the primary indicator was a high number of connection attempts originating from the same source IP address to multiple destination ports. For the repeated authentication scenario, the analysis focused on the frequency and temporal distribution of connections that target the SSH service.

In the potential data exfiltration scenario, the primary metric was the volume of data transferred. Detection was based on identifying unusually large data transfers relative to the baseline network communication observed within the environment. Given the exploratory nature of this study, no detection thresholds were defined. Instead, the evaluation relied on comparing network traffic patterns recorded during normal system operation with those generated by experimental attack scenarios.

### *5.2. Port Scanning Analysis*

To identify port scanning, an aggregation-based analysis was performed, focusing on the number of connections and the number of unique destination ports observed for all source–destination IP address pairs, without prior knowledge of the communication origin or target.

Based on the query results, a list of IP address pairs that exhibit the highest number of unique destination ports was drawn up, serving as a key indicator of potential port scanning activity (Tab. 1).

The results indicate the presence of several IP address pairs generating traffic with a clearly anomalous pattern. In particular, one pair stands out, exhibiting more than one hundred unique destination ports, which distinctly distinguishes it from typical application traffic, which is usually limited to a small number of ports.

Based on the query results, a list of IP address pairs generating the highest number of connections to the SSH service was compiled (Tab. 2).

The results indicate a significant variation in the intensity of SSH traffic between individual IP address pairs. The highest number of connections was observed for an internal address pair, with 240 connections recorded to port 22. This volume by far exceeds typical administrative traffic and may indicate repeated authentication attempts occurring within a short time window.

The remaining records include both traffic between internal resources and connections initiated from an external IP address. In these cases, the number of connections is relatively low and can be interpreted as either isolated access attempts or elements of automated scanning of publicly available services.

Traffic on ports 443 and 80 corresponds to standard network communication over HTTPS and HTTP protocols, respectively. Their presence is typical for cloud environments and does not show the regular patterns characteristic of beacon-type communication. Port 8080, despite a relatively low number of connections, was identified as a potential candidate for further analysis due to its atypical usage and previous observations suggesting its involvement in automated communication. Unlike standard system ports, non-standard ports are more frequently used as communication channels in beaconing scenarios.

A high number of repeated connections to a single port within a short period of time is a characteristic indicator of brute-force authentication attempts. The case analyzed demonstrates a clear deviation from standard administrative traffic, which is typically characterized by a limited number of sessions of longer duration. However, in cloud environments, similar traffic patterns may also be generated by automation systems or infrastructure management tools, which further complicates the unambiguous interpretation of the results.

**Tab. 1.** Results of port scanning analysis.

| SRC IP | DST IP | Connections | Dst. Ports |
|---|---|---|---|
| 10.0.1.254 | 10.0.1.180 | 257 | 122 |
| 185.125.190.58 | 10.0.1.254 | 51 | 51 |
| 91.189.91.157 | 10.0.1.254 | 45 | 45 |
| 185.125.190.57 | 10.0.1.254 | 43 | 43 |
| 185.125.190.56 | 10.0.1.180 | 42 | 42 |
| 91.189.91.157 | 10.0.1.180 | 32 | 32 |
| 194.58.204.20 | 10.0.1.180 | 31 | 31 |
| 172.232.146.46 | 10.0.1.254 | 31 | 31 |
| 45.83.221.52 | 10.0.1.180 | 30 | 30 |
| 185.125.190.56 | 10.0.1.180 | 29 | 29 |

**Tab. 2.** Analysis of SSH connection counts.

| SRC IP | DST IP | SSH connections |
|---|---|---|
| 10.0.1.180 | 10.0.1.254 | 240 |
| 83.27.3.170 | 10.0.1.254 | 12 |
| 83.27.3.170 | 10.0.1.180 | 11 |
| 10.0.1.254 | 10.0.1.180 | 7 |

### *5.3. Data Exfiltration Analysis*

Data exfiltration is characterized by an increased volume of outbound traffic from the analyzed resource, often occurring in a manner that deviates from the typical communication profile. Unlike port scanning or beaconing-type communication, the primary indicator of potential exfiltration is not the number of connections but the total amount of data transferred in a given direction. To identify potential data transfers, an analysis was performed aggregating the total number of bytes transferred and the number of flows for each destination port, without any prior knowledge of the roles of the source and destination in the communication.

In analyzing data exfiltration, the critical factor is the total volume of transmitted data, rather than the number of connections. Elevated outbound traffic can indicate a potential security breach. However, its interpretation requires considering the operational context of the system.

In practice, detecting such events solely from network traffic metadata may yield ambiguous conclusions, particularly in environments with highly variable workload profiles. However, it is important to note that an increased volume of transmitted data does not necessarily indicate a security incident.

Similar traffic patterns may also occur during legitimate operations such as backup processes, system updates, or administrative file transfers. Consequently, analyses based solely on network flow metadata may yield ambiguous results and should be supplemented with context from other data sources to improve detection accuracy.

## 6. Summary and Future Work

The objective of this study was to design an analytical procedure and to analyze network traffic in an IaaS cloud environment using network flow logs. Within the scope of the work, a research environment was created in the cloud, data collection mechanisms were configured, and selected scenarios generating characteristic network traffic patterns were analyzed.

The experiments conducted in this study were qualitative in nature and focused on identifying network traffic patterns associated with selected categories of security threats. Consequently, classification metrics such as precision, recall, and false positive rate were not evaluated, as these measures are more appropriate for approaches based on formal classification techniques or ML models.

Nevertheless, the findings clearly demonstrate that AWS VPC Flow Logs serve as a valuable source of information for the preliminary detection of anomalous activities in IaaS environments. Furthermore, the results confirm that events identified through flow log analysis should be further investigated using additional telemetry sources to allow accurate incident assessment and validation.

The analysis conducted confirmed that network flow logs enable effective anomaly detection based on traffic metadata, such as the number of connections, the number of unique destination ports, the volume of transferred data, and the number of flows. In particular, patterns characteristic of port scanning, repeated SSH connections, and potential high-volume data transfers were observed. The analysis also revealed that publicly exposed resources are subject to automated scanning by external systems almost immediately after deployment.

This phenomenon underscores the necessity of continuous network traffic monitoring, even in small and relatively simple cloud environments. Furthermore, the study highlighted the limitations of relying solely on network traffic metadata.

Based solely on flow logs, it is not possible to unambiguously distinguish between legitimate operations, such as system updates or administrative file transfers, and potential attempts at data exfiltration. This underlines the need to treat network traffic analysis as an early warning mechanism that requires further correlation with additional information sources.

Overall, the work demonstrated that the proposed approach allows for the effective detection of atypical communication patterns and provides a solid foundation for the further development of security monitoring systems. In the future, the analysis could be extended to include automatic event classification, correlation with system logs, and application of ML techniques to reduce false positives.

The analysis indicates that even under limited data visibility, it is possible to identify significant deviations from normal network behavior. The results highlight the need to adopt a multisource approach, in which the analysis of network flow logs is only one component of a broader security monitoring system.

Future work will extend this research by incorporating a quantitative evaluation of detection performance, including metrics such as detection rate, false positive rate, and detection time. Another promising direction is the comparative evaluation of anomaly detection using AWS VPC Flow Logs against the native security capabilities within the AWS ecosystem.

## References


[1] S. Kanthed, "Monitoring of Cloud Computing Environments: Concepts, Solutions, Trends, and Future Directions", *International Journal on Science and Technology*, vol. 15, pp. 1–16, 2024 (https://doi.org/10.71097/IJSAT.v15.i1.2836).

[2] A. Mycek, "Monitoring, Management, and Analysis of Security Aspects of IaaS Environments", *Journal of Telecommunications and Information Technology*, vol. 94, pp. 108–116, 2023 (https://doi.org/10.26636/jtit.2023.4.1419).

[3] S. Shetty, B. Biswal, and H. Maziku, "Auditing and Analysis of Network Traffic in Cloud Environment", *International Journal of Business Process Integration and Management*, vol. 7, pp. 153–165, 2014 (https://doi.org/10.1504/IJBPIM.2014.063519).

[4] B. Thomas *et al.*, "A Study of Cloud-native Intrusion Detection Using VPC Flow Logs and Ensemble Learning", *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 15, pp. 985–993, 2026 (https://doi.org/10.17148/IJARCCE.2026.151131).

[5] H.B. Illa, "AI-Driven Incident Detection Using AWS CloudWatch and VPC Flow Logs", *South Asian Journal of Engineering and Technology*, vol. 14, pp. 70–85, 2024 (https://doi.org/10.26524/sajet.2024.14.32).

[6] M. Collins *et al.*, "Superflows: A New Tool for Forensic Network Flow Analysis", *arXiv*, 2024 (https://doi.org/10.48550/arXiv.2403.01314).

[7] K. Hsieh *et al.*, "NetVigil: Robust and Low-cost Anomaly Detection for East-West Data Center Security", *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 1771–1789, 2024.

[8] Z. Wang *et al.*, "Diagnosing Application-network Anomalies for Millions of IPs in Production Clouds", *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pp. 885–899, 2024.

[9] M.S. Islam *et al.*, "Anomaly Detection in Large-scale Cloud Systems: An Industry Case and Dataset", *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Ottawa, Canada, 2025 (https://doi.org/10.1109/ICSE-SEIP66354.2025.00039).

[10] J. Kalibjian, "Security Considerations When Performing Telemetry Post-processing and Analysis in Cloud Environments", *International Telemetering Conference Proceedings*, vol. 58, 2023.

[11] W. Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a Service Security: Challenges and Solutions", *2010 The 7th International Conference on Informatics and Systems (INFOS)*, Cairo, Egypt, 2010.

[12] F. Minna *et al.*, "Understanding the Security Implications of Kubernetes Networking", *IEEE Security & Privacy*, vol. 19, pp. 46–56, 2021 (https://doi.org/10.1109/MSEC.2021.3094726).

[13] A. Mycek and M. Łukaczyk, "Security and Hardening of Kubernetes in Public Clouds: A Comparative Study of EKS, AKS, and GKE", *IEEE Access*, vol. 14, pp. 80990–81009, 2026 (https://doi.org/10.1109/ACCESS.2026.3697610).

[14] T.A.K. Manne, "Leveraging AWS Security Hub and GuardDuty for Continuous Threat Intelligence", *The Journal of Scientific and Engineering Research*, vol. 11, pp. 268–275, 2024 (https://doi.org/10.5281/zenodo.17199795).

[15] A. Mycek, D. Grzonka, and J. Tchórzewski, "Agent-based Simulation and Analysis of Infrastructure-as-Code Process to Build and Manage Cloud Environment", *Proc. of the 37th ECMS International Conference on Modelling and Simulation*, pp. 513–520, 2023 (https://doi.org/10.7148/2023-0513).

[16] A. Mycek and M. Łukaczyk, "Security of Containerization Platforms: Threat Modelling, Vulnerability Analysis, and Risk Mitigation", *Proc. of the 38th ECMS International Conference on Modelling and Simulation*, pp. 585–591, 2024 (https://doi.org/10.7148/2024-0585).

[17] M. Łukaczyk and A. Mycek, "Automation of Security Policies in DevSecOps Environments: Implementation of Zero Trust Principles Using Ansible and Terraform in Linux Systems", *Proc. of the 39th ECMS International Conference on Modelling and Simulation*, pp. 241–247, 2025 (https://doi.org/10.7148/2025-0241).

**Hubert Wójcik, B.Eng.**
Department of Computer Science
E-mail: hubert.wojcik78@student.pk.edu.pl
Cracow University of Technology, Cracow, Poland
https://www.pk.edu.pl

**Mirosław Roszkowski, M.Sc.**
CUT Doctoral School, Cracow, Poland
https://orcid.org/0009-0008-4464-9430
E-mail: miroslaw.roszkowski@pk.edu.pl
Cracow University of Technology, Cracow, Poland
https://www.pk.edu.pl

**Andrzej Mycek, M.Sc.**
CUT Doctoral School, Cracow, Poland
https://orcid.org/0000-0002-2720-6071
E-mail: andrzej.mycek@pk.edu.pl
Cracow University of Technology, Cracow, Poland
https://www.pk.edu.pl