WHITE PAPER

Ebb and Flow: Network Flow Logging as a Staple of Public Cloud Visibility or a Waning Imperative?

Dennis Taggart

Ebb and Flow: Network Flow Logging as a Staple of Public Cloud Visibility or a Waning Imperative?

GIAC (GPEN) Gold Certification

Author: Dennis Taggart, d3soteric@icloud.com
Advisor: Jonathan Risto

Accepted: 4/16/2020

Abstract

The basic tenets of information security remain relatively unchanged even while specific examples of security-related tools, processes, and procedures may shift in popularity over time. Deciding what to prioritize and recommend as a security professional can be challenging, but the most straightforward cases are those justified by the quantitative reduction of risk. In this search for quantitative risk reduction, it is worthwhile for security professionals to consider that the methods used to fulfill basic security needs in one environment may not provide the same benefit in another. The 2019 version of the Cloud Security Alliance's *Top Threats to Cloud Computing* document warns of critical security issues facing public cloud consumers (Cloud Security Alliance, 2019, p.40). The CSA also acknowledges their work concentrates less on some of the more traditional security threats like "vulnerabilities and malware", while calling for further research (Cloud Security Alliance, 2019, p.40). This whitepaper inhabits the category of additional research and also occupies a space parallel, but perhaps not identical to classical security views. This research assumes a slightly-less-traditional approach by not taking the value of flow logging, or its costs in the cloud, for granted. It further asserts that given limited resources, there may be more directly valuable logging sources available. This paper establishes a quantitative methodology for judging the effectiveness of flow and non-flow logging as applied in a public cloud environment. It exercises this methodology by simulating top cloud computing threats and examining the capabilities of each.

1. Introduction

Security-related tools, processes and procedures may change over time. However, the core objectives of information security remain relatively static. Yet, when presented with competing security objectives, prioritization can be difficult. Despite the challenge, the simplest security proposals to justify are those based on quantitative risk reduction. While seeking a measurable reduction in risk in a new environment, even time-proven security methods deserve scrutiny, as they may not provide the same benefit in a new environment.

The shared responsibility model makes some of this obvious. An IaaS customer does not physically secure their cloud service provider's (CSP) datacenter, but the utilization of an IaaS provider's services does not mean that physical security is no longer relevant, yet does indicate it may have changed. The logging of network traffic presents a similar case. On the one hand, systems can be deployed as code to simply restrict access to approved sources and non-network logs may give visibility into peculiarities. On the other hand, network traffic patterns from flow logs, which lack application data, may still give early views into the unexpected. Because network flow logging is often regarded as a staple of security visibility (Dickinson 2019; Szili, 2019), this research concentrates on establishing the value of flow logging in public cloud environments.

The core purpose of the paper is twofold: educate security practitioners regarding the costs associated with flow logging in cloud environments, and test the effectiveness of flow logging concerning specific public cloud threats. The researcher provides precise data with cloud-relevant examples to aid in value-driven security decisions regarding flow logging. Clear data will enable security professionals to decide if the flow logging investment is right for their organization. This data may also be useful to guide discussion around organizational monitoring strategy in public clouds.

Flow logging provides metadata regarding network layer communications such as source and destination and has been described succinctly as providing a "high-level view of network traffic" (Dickinson, 2019, p. 5). Flow logs include information such as source and destination IP addresses, port, and protocol, as seen in Figure 1.

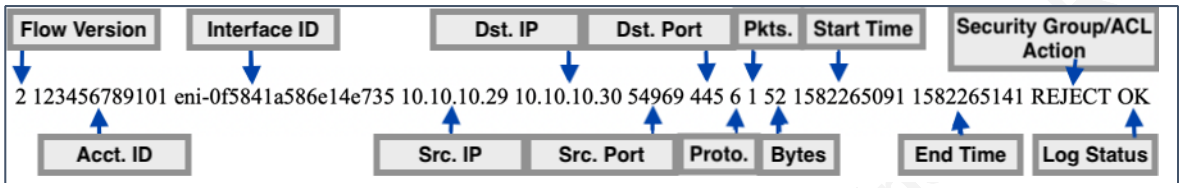

Flow Version
Interface ID
Dst. IP
Dst. Port
Pkts.
Start Time
Security Group/ACL Action
2 123456789101 eni-0f5841a586e14e735 10.10.10.29 10.10.10.30 54969 445 6 1 52 1582265091 1582265141 REJECT OK
Acct. ID
Src. IP
Src. Port
Proto.
Bytes
End Time
Log Status

Figure 1: Depiction of AWS Flow Log Format containing default fields described on (Amazon, n.d. a)

Because flow logging does not include the packet data, individually, these logs do not require much space. Add the importance of visibility to information security and the streamlined implementations to enable these logs offered by major cloud providers, and a clearer picture begins to form around why flow logs are often looked upon favorably.

Although there are enticing facets to flow logging in the cloud, some of its limitations have been documented (Dugan, 2019). The total cost of ownership related to flow logging can also be high. Amazon's pricing example for flow logging services (referred to as vended logs) estimates a bill of over $13,000/mo. for 72TB of logs/mo. (Amazon, n.d. c). If that seems expensive, Amazon offers a discount of up to 50% per GB (Amazon, n.d. c), depending on volume, if the flow logs are configured to bypass CloudWatch and instead are sent directly to Amazon's Simple Storage Service (S3). This means that the cost to capture 72TB of flow logs and store them for a month could be reduced to $8,294/mo. (Amazon, n.d. c). Microsoft also warns of potentially high costs (Microsoft, 2017) and does not publicly offer a tiered pricing model for flow logs in their Network Watcher service (Microsoft, n.d.). Google Cloud Platform (GCP) offers a tiered pricing model like Amazon's (Google, n.d. a), but if logs are sent to Cloud Logging, then the tiers disappear, leading to charges more consistent with Azure (Google, n.d. b). Neither Google nor Microsoft currently advertises direct-to-storage discounts. Even when organizations consider markdowns, if they generate large enough quantities of flow logs, the costs could be prohibitive. Aversion to high fees is conspicuously typical for not-for-profits or small and medium-sized businesses conducting large network operations. Adding to the conundrum is the fact that it is difficult to know how many logs will be generated until they are created. If flow logging is to be enabled, then security practitioners need to be prepared to speak to their value.

The concern that transplanting previous modes of security at a premium into an environment where these methods are less efficient is not unwarranted. Even as the dynamic nature of cloud computing is acknowledged to complicate the implementation of an intrusion detection system (IDS), some have employed sophisticated methods in an attempt to keep up with the rapid changes in cloud environments (Li et al., 2017). Others forecasted a vital role for Network IDS (NIDS) in mitigating cloud threats such as denial of service (DoS) attacks (Gul and Hussein, 2011). Because resilience to DoS attacks became a de-facto capability of the cloud, the need for customers to fill that role with tooling such as NIDS was diminished. While network or operating system (OS) logs have been go-to sources for intrusion monitoring (Kumar, 2014), cloud environments offer log sources such as management APIs or statistics, which may provide a more directly positive impact at a lower cost.

Beyond establishing the natural propensity to retain familiar security methods established above, another foundational premise of this research is that there is an asymmetric relationship between enabling services and optimizing them. To illustrate this notion, consider SIEM management for a moment. It is reasonable to expect an assigned employee to spend upwards of 80% of their time analyzing and reviewing alert logic (Hubbard et al., 2019). It is less reasonable to expect to spend 80% of one's time solely enabling new data sources for the SIEM. Once logs are collected, a significant portion of work remains which corroborates the claim of an asymmetric relationship. Infuse the pay-per-use model of public cloud environments into log source collection and management, and it is clear how enabling new logging requires careful consideration to both achieve value and avoid ballooning costs.

2. Research Method

The environment for this research is built on Amazon Web Services (AWS). High-level comparisons across cloud service providers (CSPs) are provided. Any comparison here is not exhaustive, and all tests were performed in AWS. Three AWS virtual private clouds (VPCs) with two separate AWS accounts were created to facilitate multiple configurations, supporting the assessment of several scenarios as needed in a controlled setting. Where no difference in functionality was expected or observed, a single VPC or account was used in testing. Flow logs are compared to non-flow records and the findings provided by an AWS monitoring service: GuardDuty. A description of GuardDuty and similar services is supplied in section 3.1.3.

VPC	**Test**	**Control**	**Comparison**
Purpose	Flow Logging Utilized	Non-flow Logging Utilized	GuardDuty Utilized

Table 1: Overview of test environment and intended purposes for each.

This whitepaper examines the contribution of flow logging in detecting relevant cloud attacks. The relevance of attacks was determined based on threats included in *The Cloud Security Alliance's Egregious 11: Cloud Computing Top Threats* document (Cloud Security Alliance, 2019). Other research has been performed on earlier versions of the Top Threats document (Wongthai et al., 2013), which helps fortify its position as a valuable structure to base research on. For this paper, the threats were grouped, and simulations to embody those threats were assigned. A representation of these groupings and a description of the simulation is provided in the next subsection within Table 2. Grouping threats is justified for this research because a single simulation may have multiple threat applications. Importantly, the researcher designed the simulations to comply with the boundaries set in the AWS penetration testing policy (Amazon, n.d. b).

2.1. Foundation of Standard Used

Inclusion in *The CSA Egregious 11* was chosen as the basis for organizing the simulations for five primary reasons. It is regularly updated, cloud-focused, has earned wide recognition, is freely available, and is community-driven. The groupings in table 2 do not indicate that there are no other ways to arrange them. Each of the CSA top threats will be covered in their groups from the perspectives of non-flow, flow, and the GuardDuty service in section 3.

CSA Threat(s)	Simulation
Lack of Cloud Security Architecture and Strategy (3); Limited Cloud Usage Visibility (10)	Troubleshooting infrastructure configuration or other failures.
Metastructure and Applistructure Failures (9)	An attacker attempting to pollute flow logs with misleading data e.g. false source/destination information to distract analysts or tools.
Misconfig. and Inadequate Change Control (2); Weak Control Plane (8); Abuse and Nefarious Use of Cloud Services (11)	An attacker attempting to download/upload over-exposed data in cloud storage solutions (S3, RDS, ElasticSearch).
Insufficient Identity, Credential, Access and Key Management (4); Account Hijacking (5); Insider Threat (6)	An attacker who has acquired access tokens and is abusing them to abuse or launch other services within the targeted environment.
Data Breaches (1); Insecure Interfaces and APIs (7)	An attacker engaging in application layer attacks such as SQL Injection.

Table 2: List of CSA threat groupings and brief descriptions of intended simulations.

3. Findings, Examples and Discussion

This section describes the research performed according to the methodology introduced above. Note that flow logging is available from each of the big three CSPs (Amazon, Google and Microsoft). Although flow logs are created for each host communicating on the network in every environment, the location for commissioning this logging varies slightly between providers. AWS allows the most flexibility, obliging customers to enable the flow logs for a single elastic network interface (ENI), all interfaces in a subnet, or at the VPC level (Amazon, 2020a). GCP allows enabling flow logging for all hosts in a subnet (Google, 2020). Azure flow logging is toggled at the network security group (NSG) and these groups can be associated on an individual VM basis, or to an entire subnet (Microsoft, 2017).

3.1. Lack of Cloud Security Architecture and Strategy & Limited Cloud Usage Visibility

A common use case for enabling flow logging is to enhance network troubleshooting capability. The test environment's infrastructure was created through a CloudFormation template (Taggart, 2020). Limited cloud usage visibility and a lack of cloud security architecture are simulated here through the troubleshooting of the non-functional network.

At first, the testing environment as created by the template seemed to be functioning normally, however when instances were created, they were not accessible. Omitting considerations of security architecture and opening all ports on the security group (SG) did not resolve the issue.

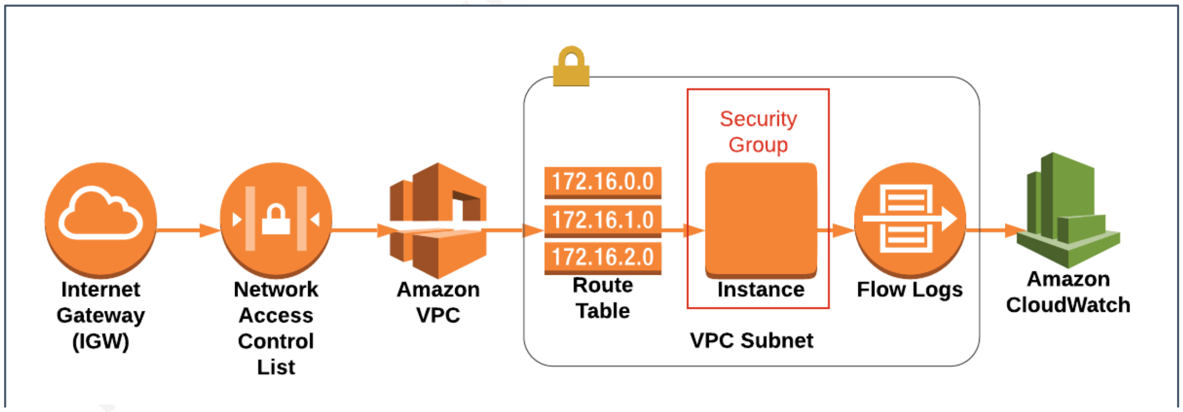


Figure 2: Diagram of Potential Architectural Problem Areas

Four tests were conducted to understand the problem. The issue was found to exist in two misconfigurations. Referencing Figure 2, the SG only allowed traffic to other devices with the same SG attached. There was another problem in that the route which was being created to allow traffic to the internet was never successfully connected to the subnet. This prevented all network access. Although this issue was first recognized by performing a code review and reading the AWS documentation, such a white/crystal box approach may not be possible for one with little cloud infrastructure experience or with limited access to view the configuration files. This limited access to infrastructure and code furnishes an example of how flow logging can be beneficial.

3.1.1. Utility of non-flow logs

Without flow logs, data useful to troubleshooting included architecture diagrams, AWS documentation, and the CloudFormation template used to create the infrastructure. CloudTrail logging showed that the API calls for CreateSubnet, CreateRouteTable, and CreateRoute commands were each created successfully. Analysts not knowing that the AssociateRouteTable API call is required to tie the resources together would be unlikely to catch this misconfiguration through CloudTrail logs alone. Gaining a better understanding of the necessary API calls was ultimately all that was needed to solve these issues. The CloudTrails did not require CloudFormation access and could be made accessible to analysts for troubleshooting purposes.

3.1.2. Utility of flow logs

	Test 1	**Test 2**	**Test 3**	**Test 4**
Security Group Configuration	Default (VPC Only)	Open to Anywhere	Default (VPC Only)	Open to Anywhere
Route Table Configuration	Default (VPC Only)	Default (VPC Only)	Open to Anywhere	Open to Anywhere
Flow Log Data	REJECT (all)	ACCEPT (in) REJECT (out)	REJECT (in) ACCEPT (out)	ACCEPT (all)
Ping Data	Timeout	Timeout	Timeout	Response

Table 3: Summary data from troubleshooting security group and general architecture issues.

Table 3 describes the configurations through which traffic was sent to troubleshoot unknown network issues. Although in Test 2, the external client never received a response for the traffic sent, the network interface was observed to be receiving the packets. Tests 2 and 3 illustrated that it may be necessary to enable flow in two accounts to gain the entire picture across multiple environments. Successful outbound NTP traffic and the correlated responses were detected during Test 3. This was a great indication that underlying network issues had been resolved and that the SG required additional attention. Flow logs provided a means to differentiate between different problems, which further aided troubleshooting.

3.1.3. Utility of GuardDuty

Some cloud service providers (CSPs) offer threat detection services that perform activities such as behavior analytics, anomaly detection, and threat feed monitoring to alert on suspicious activity, requiring little configuration. Abstracted monitoring services like these are particularly applicable here because they monitor network flows, among other data sources, but do not require that flow logs are enabled separately or paid for directly by the customer.

The name of the service AWS offers in this space is GuardDuty. It monitors DNS logs, API activity, and VPC flow logs to detect and alert on anomalous activity and charges by the aggregate size in GB of records and the number of CloudTrail events analyzed (Amazon, n.d. d). The other side of this arrangement is the data analyzed by the service is not available for analysis outside of GuardDuty and its detections. Findings are limited to those that the service offers. Microsoft provides a similar service through its Azure Security Center threat protection (Microsoft, 2019a). GCP has no comparable threat detection service available at the time of writing.

Amazon GuardDuty relies on observed data and threat lists to detect potentially malicious activity. Because the CSA cloud threats analyzed here: a lack of security architecture and strategy, and limited cloud usage visibility are both issues of omission, these threats are more likely themselves to lead to an absence of logging or logging without accompanying analysis. No GuardDuty alerts were expected for this simulation. Note that leaving SSH or RDP open to the public internet could indicate a lack of security architecture and strategy. GuardDuty is able to detect brute-force attempts against these protocols if they are configured to use the standard ports.

3.2. Metastructure and Applistructure Failures

Metastructure refers to the infrastructure supporting the control plane provided by the cloud service provider, whereas the applistructure describes the software components from which the application stack is comprised (Cloud Security Alliance, 2019). Security failures in either of these layers are inherently difficult to detect because they can reside in unknown, unexpected, or abstracted areas. For example, entire sections of the metastructure layer may be opaque to the customer. Similarly, applistructure issues may exist as previously unknown vulnerabilities in an imported library, machine images (Cloud Security Alliance, 2019) or an upstream API provided by an external data-partner. Given the unknown nature of issues leading to security failures related to this top threat, if these systems utilize network access to function, flow logging can assist in detecting weaknesses. To simulate metastructure security failures and measure the effectiveness of flow logging, this simulation examined instances sending crafted packets using scapy. Figure 3 diagrams how the test spanned multiple VPCs.

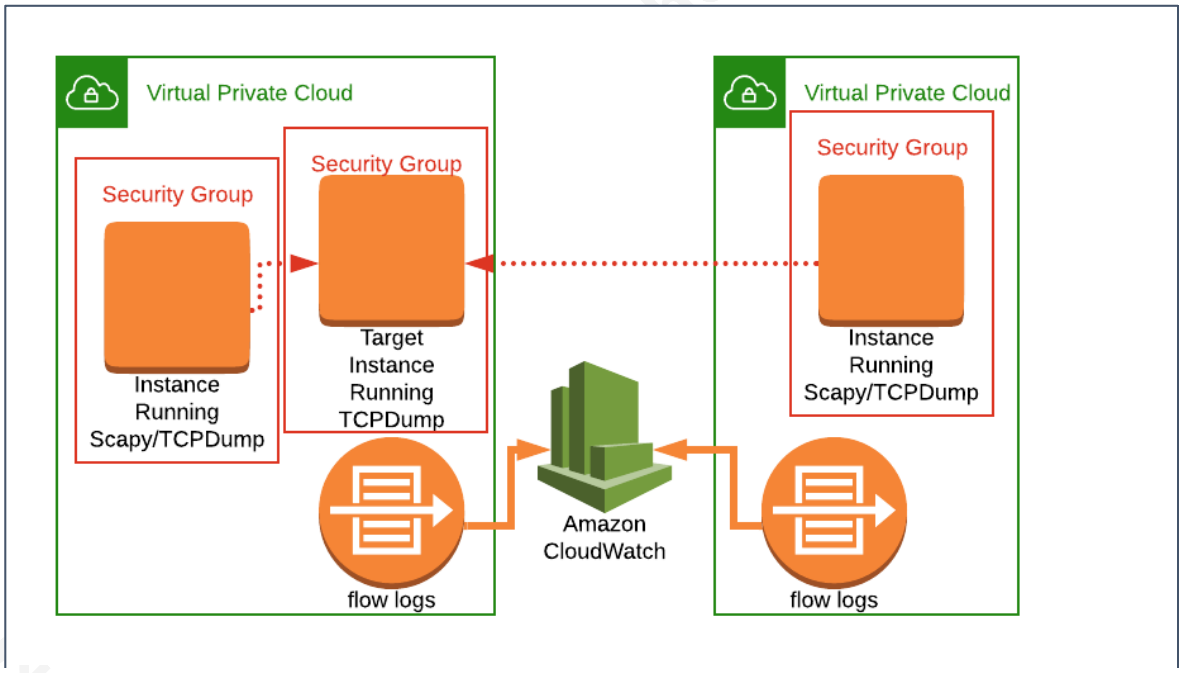


Figure 3: Diagram of systems used for applistructure/metastructure simulation. Each instance was configured to use flow logging.

3.2.1. Utility of non-flow logs

It is possible to use native tools such as CloudWatch alarms to create simple alerts for unusual changes in resource consumption. For example, if a particular system is not expected to make many outbound network connections, changes in the number of packets or bytes sent could encourage additional investigation, even without details such as destination IP address or port as provided in flow logging. Native tools such as tcpdump can assist with gathering network information without flow, but these may not scale well.

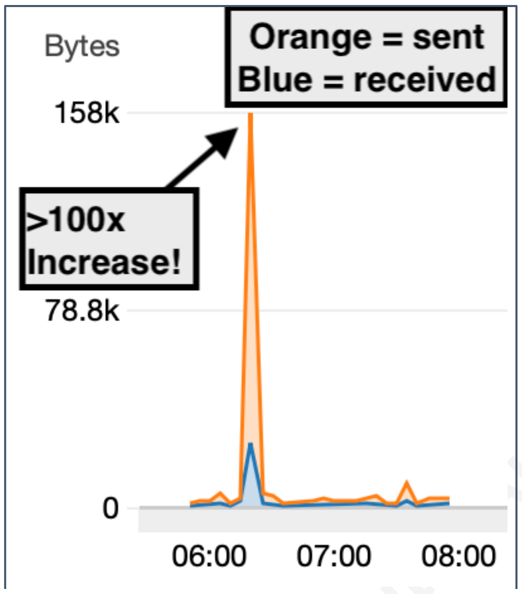


Figure 4: CloudWatch dashboard illustrating a change in outbound network packets relative to a baseline.

3.2.2. Utility of flow logs

AWS VPCs have documented security controls in place to prevent spoofing of network traffic. EC2 Instances are expressly prohibited from sending any traffic from IP or MAC addresses not assigned to them (Amazon, 2016). This was validated as accurate by observing the spoofed traffic egress, as detected via tcpdump, and noting that the packet never arrived at the intended ingress interface. No flow logs were generated for these spoofed transactions.

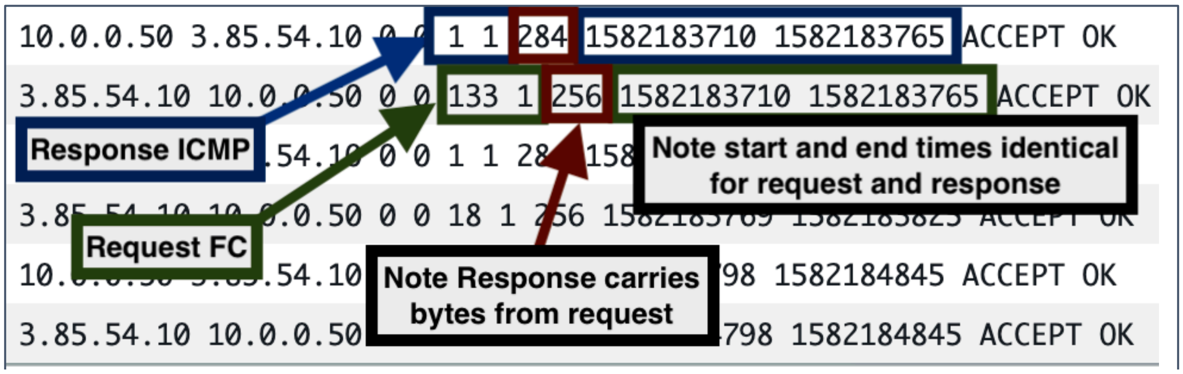
When spoofing IP protocol values, the security group uses the protocol value specified in the IP layer to determine which protocol the security group will apply to. This affected the packet contents included in the flow log. If the protocol specified was 1 (ICMP), but TCP headers (protocol 6) such as source or destination port were included, the flow log did not include those additional headers.

One of the more interesting discoveries from this research was that if a protocol which the instance does not support is allowed on an inbound security group (SG), it is possible to generate outbound ICMP "protocol not supported" messages. Even if all SG rules allowing outbound traffic are removed from the SG attached to the instance, this traffic is sent. This is noteworthy because SGs operate with an implicit deny model. An example scapy command is: `send(IP(dst="<target>", proto=134)/IP())`.

One way this type of condition could be abused is if a malicious actor configured a large number of small instances, then opened various protocols in the SG, with the intent to allow accept a large number of spoofed packets. This would in turn reflect these ICMP packets to arbitrary locations, enabling an attempt to deny service for a target. Note that no DoS attacks were attempted.

Seven of 255 protocol numbers (IANA, 2020) tested did not result in an ICMP response: specifically, protocols 2, 6, 17, 41, 58, 103, 136. While some protocol support was expected, the count was unknown. It was further expected that any ICMP traffic, being a different protocol than the request, would be blocked by the SG. Even though SGs were understood to be stateful, the way some protocols did not lead to ICMP messages seemed to complicate the issue, so additional explanation was needed.

Given the capacity for malicious application, a potential vulnerability report was filed with AWS according to their processes (Amazon, n.d. e). After some rounds of coordination and additional research, it was determined that the SGs are working as intended and that ICMP messages resulting from allowed traffic pass through even locked-down security groups as part of the SG's state tracking. The related work was not fruitless, however, as this exercise revealed there appears to be support included for seven protocols in the default Amazon Linux 2 amazon machine image (AMI), otherwise protocol unsupported messages would be expected for these just as with the other protocols. This is a good example of how metastructure idiosyncrasies can affect customers and exemplifies how flow logs can be used to guide deeper investigation into esoteric subject areas.



```
10.0.0.50 3.85.54.10 0 0 1 1 284 1582183710 1582183765 ACCEPT OK
3.85.54.10 10.0.0.50 0 0 133 1 256 1582183710 1582183765 ACCEPT OK
.54.10 0 0 1 1 28 158
3.85.54.10 10.0.0.50 0 0 18 1 256 1582183769 1582183825 ACCEPT OK
10.          .54.10                   98 1582184845 ACCEPT OK
3.85.54.10 10.0.0.50                 798 1582184845 ACCEPT OK
```

Response ICMP
Request FC
Note start and end times identical for request and response
Note Response carries bytes from request

Figure 5: Six flow logs illustrating the pattern of when an unsupported protocol was encountered, an ICMP protocol unreachable message was successfully sent, even though no applicable outbound traffic rule was in the SG.

On the applistructure level, flow logs can be used to detect unexpected outbound connections, even while not allowing traffic to leave the VPC. This was demonstrated in simulation one, when the basic Amazon Linux 2 AMI attempted to make NTP connections, which created flow log entries. Because the security group was restrictive, outbound network access was denied and logged. The use of NTP on a standard AWS image is somewhat notable in itself because these instances have access to AWS' native time sync service provided by the metastructure (Amazon, 2017) and that traffic would not be recorded in the flow logs (Amazon, n.d. a).

3.2.3. Utility of GuardDuty

GuardDuty did not alert on attempted spoofing activity. For traffic dismissed outright by the AWS infrastructure as in this case, it could have been interesting to have a means of viewing traffic dropped by the infrastructure.

3.3. Misconfiguration and Inadequate Change Control; Abuse and Nefarious Use of Cloud Services; Weak Control Plane

This simulation used open S3 buckets, databases configured via Amazon's Relational Database Service (RDS), and Amazon's ElasticSearch Service to create environments where abuse of cloud storage services could be emulated. For S3, flow logs are not available. However, to promote added visibility, both server and object-level logging were enabled. PostgreSQL logs were enabled in RDS, and both RDS and ElasticSearch had flow logging enabled. Standard commands such as *cp* and *sync* for S3, or *select* for RDS were run to imitate discovery and exfiltration. Note that while RDS is explicitly approved for penetration testing, the S3 and ElasticSearch services are excluded. Therefore, only legitimate queries were performed, and the services themselves were not tested.

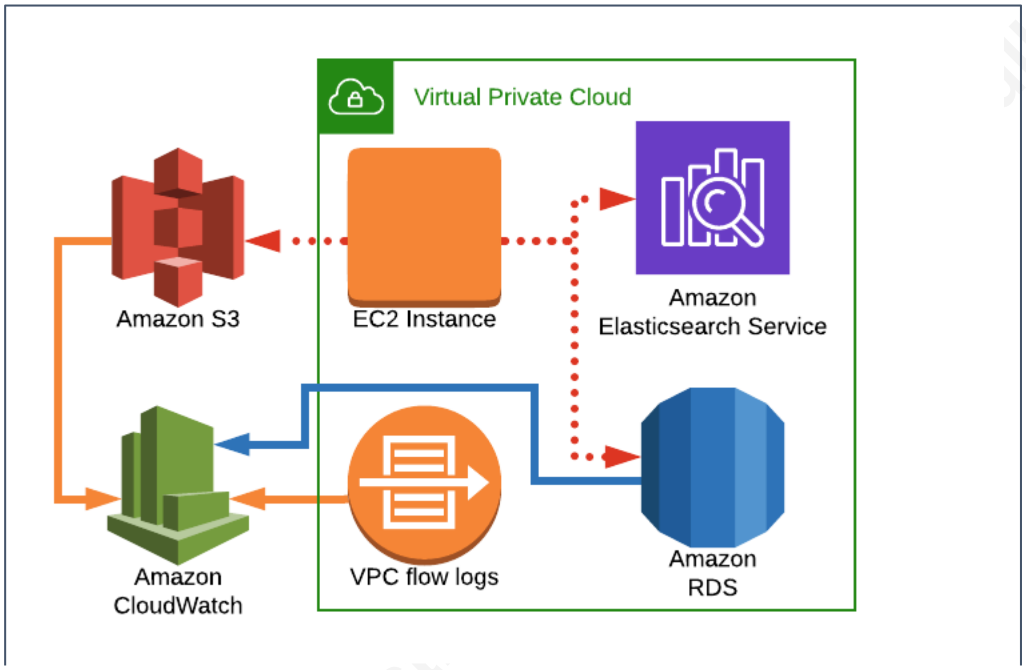
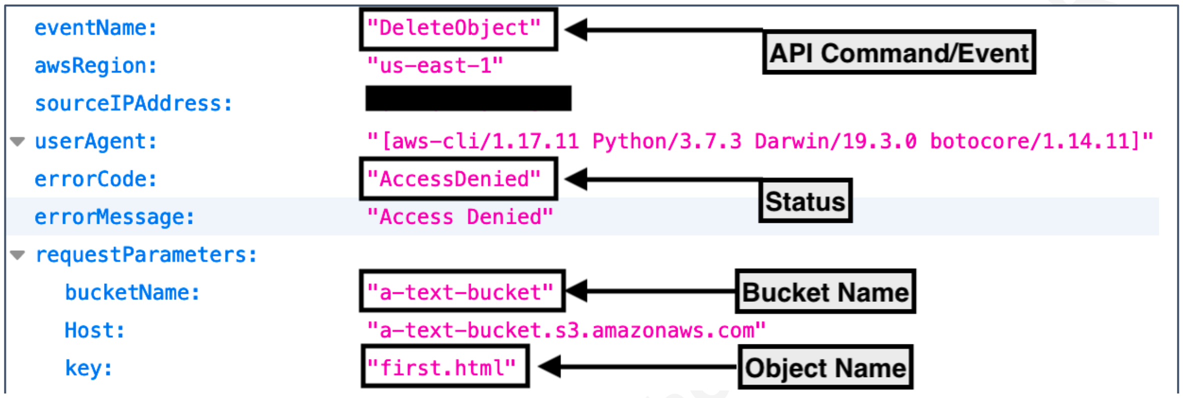


Figure 6: Architecture diagram illustrating the simulated attacker EC2 instance sending queries in an attempt to access the target data. Note that EC2, ElasticSearch and RDS are compatible with flow logging.

3.3.1. Utility of non-flow logs

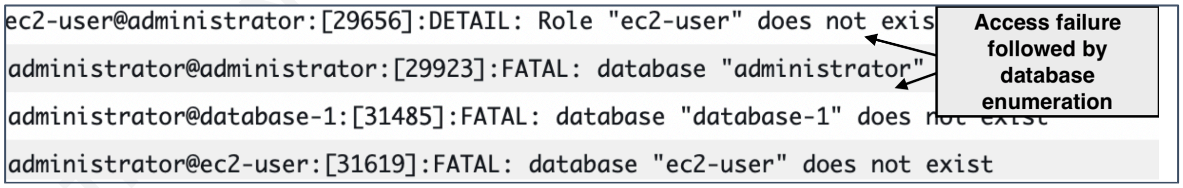
Because S3 resides outside of the VPC environment and so is unable to utilize flow logs, other sources must be drawn upon for visibility. Server-level logging provides detail around API commands run against the bucket, such as deleting the bucket policy with REST.DELETE.BUCKETPOLICY. In addition to control plane logs, object-level logging is available for an added expense and can provides detail more directly applicable for a security practitioner. Source IP addresses, commands run (eventName), and which object was accessed (key) are all available, as depicted in figure 7. If a public S3 bucket were to be abused by an attacker to host malicious content, object logging would be the most direct path to finding when that content was first uploaded.



```
eventName:          "DeleteObject"
awsRegion:          "us-east-1"
sourceIPAddress:    [redacted]
userAgent:          "[aws-cli/1.17.11 Python/3.7.3 Darwin/19.3.0 botocore/1.14.11]"
errorCode:          "AccessDenied"
errorMessage:       "Access Denied"
requestParameters:
   bucketName:      "a-text-bucket"
   Host:            "a-text-bucket.s3.amazonaws.com"
   key:             "first.html"
```

Figure 7: Example S3 Object-level Log

RDS provides a mechanism to send database logs to CloudWatch. These logs can provide security insight into the types of events occurring on the database, such as invalid queries or other errors an attacker might make while acquainting themselves with a database. Because RDS abstracts the underlying server infrastructure away from the user, it isn't possible to modify the database configuration files to send *all* audit logs to CloudWatch, instead of merely failures.



```
ec2-user@administrator:[29656]:DETAIL: Role "ec2-user" does not exis
administrator@administrator:[29923]:FATAL: database "administrator"
administrator@database-1:[31485]:FATAL: database "database-1" does not exist
administrator@ec2-user:[31619]:FATAL: database "ec2-user" does not exist
```

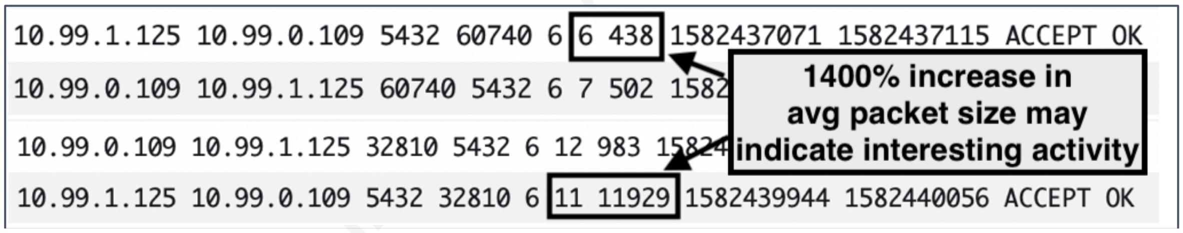
Figure 8: Example of DB failure logs showing an attacker attempt to gain access to a database.

The AWS ElasticSearch (ES) service does not expose application activity to logging, and analogous to the RDS service, ElasticSearch abstracts its underlying infrastructure away from the user. The ability to modify the application log location to view them is not currently possible. ElasticSearch does allow for control plane auditing through CloudTrail, which gives awareness to high-level changes to the ElasticSearch service. An example of this is found in logs to delete an ElasticSearch domain. While this could be indicative of someone trying to disrupt monitoring efforts, these CloudTrails are not going to be of much help to determine if an ElasticSearch domain is overexposed to the internet and leaking data.

Both RDS and ES provide examples of how the capabilities of the control plane affect the security capabilities of the services. Without the ability to customize logging, visibility into application attacks, or abuse of data is limited.

3.3.2. Utility of flow logs

Flow logging for this use case was of limited efficacy due to how most of the traffic was occurring at the application layer or was not available for the service. Flow logging could provide value in detecting unexpected external IPs connecting to RDS or ES and by providing data for analysis on the number of bytes transferred in each transaction. While raw network flows alone would rarely be sufficient to definitively demonstrate a successful application layer attack, with an understanding of healthy communication trends, these data points can provide additional context and help detect misconfigurations or overexposure.



```
10.99.1.125 10.99.0.109 5432 60740 6 6 438 1582437071 1582437115 ACCEPT OK
10.99.0.109 10.99.1.125 60740 5432 6 7 502 1582
10.99.0.109 10.99.1.125 32810 5432 6 12 983 15824
10.99.1.125 10.99.0.109 5432 32810 6 11 11929 1582439944 1582440056 ACCEPT OK
```

1400% increase in avg packet size may indicate interesting activity

Figure 9: Flow logs may provide useful metadata into application activity, even if they lack resolution for application layer data.

3.3.3. Utility of GuardDuty

Amazon GuardDuty provided various security insights even without exposing the flow logs themselves. A port scan run against an RDS database was detected along with several other GuardDuty findings around S3 policy and resource permissions. There is no finding currently available in GuardDuty to detect SQL injection (SQLi) attacks. Note that Azure's Advanced Threat Detection service for Azure SQL databases is equipped to detect SQLi (Microsoft, 2019b). There are also findings designed to detect outbound DoS or spam attacks, which could indicate nefarious use.

3.4. Insufficient Identity, Credential, and Access Management; Account Hijacking; Insider Threat

For this simulation, valid credentials were used to make changes in the testing environment. Part of this simulation's methodology was to simply use the management APIs to make changes as if unauthorized to do so and as if multi-factor authentication tokens were not enabled or were accessible by the attacker. The other part of this simulation was retrieving and utilizing an instance profile, or role assigned to an instance, to make changes in the Amazon account with the permissions attached to that system as would be performed in a Server-side Request Forgery (SSRF) attack. This type of simulation can have broad application across the top threats. For example, exploiting the access a principal has in an AWS account applies to the insufficient identity, credential, access, and key management threat. Gaining unauthorized access to a credential also applies to the account hijacking threat because, depending on the credentials gleaned, partial or near-total account takeover is possible. Finally, insider threats can abuse their access to attempt to elevate privileges or make configuration changes to avoid detection.

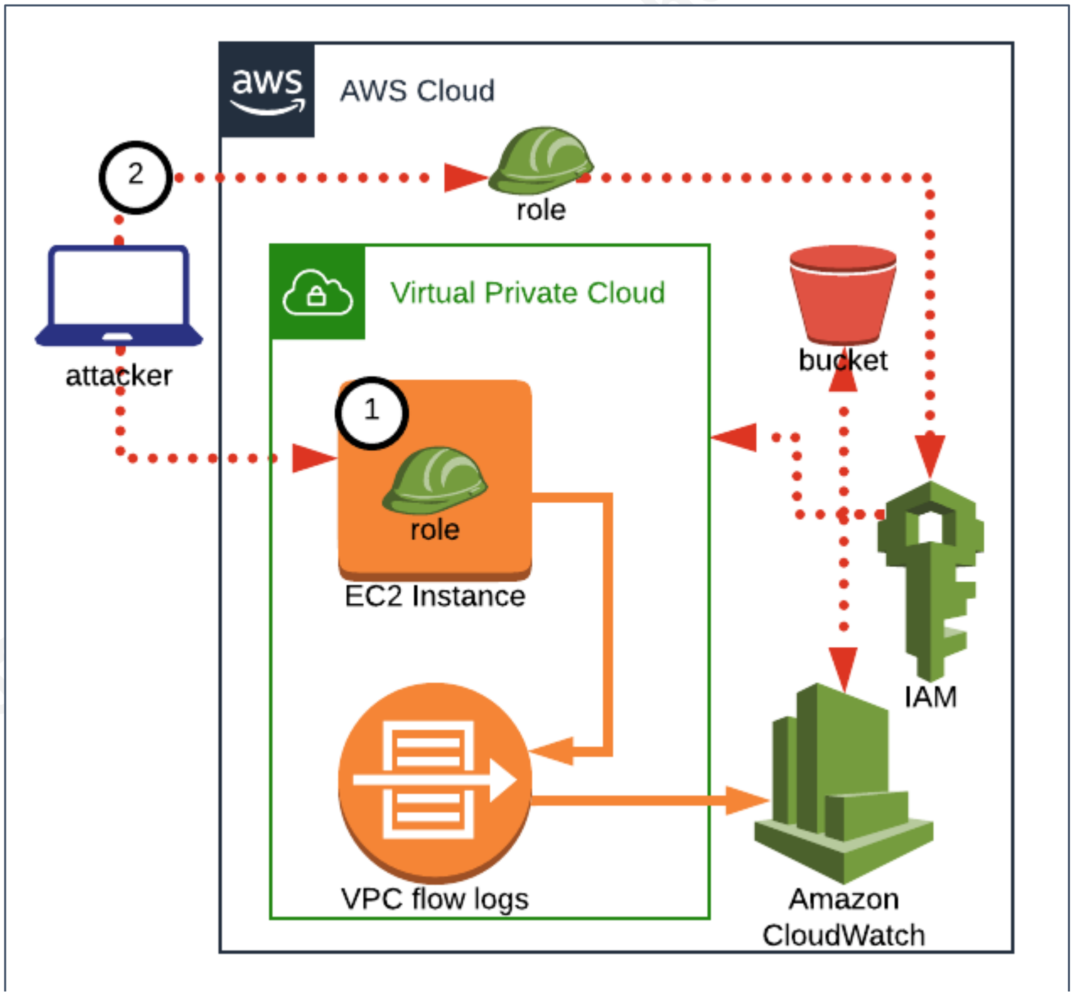


Figure 10: Illustration depicting the acquisition (1) and abuse (2) of an IAM role to modify any services the role has access to, with a minimal network footprint. The logging configuration could even be modified to alter visibility.

3.4.1. Utility of non-flow logs

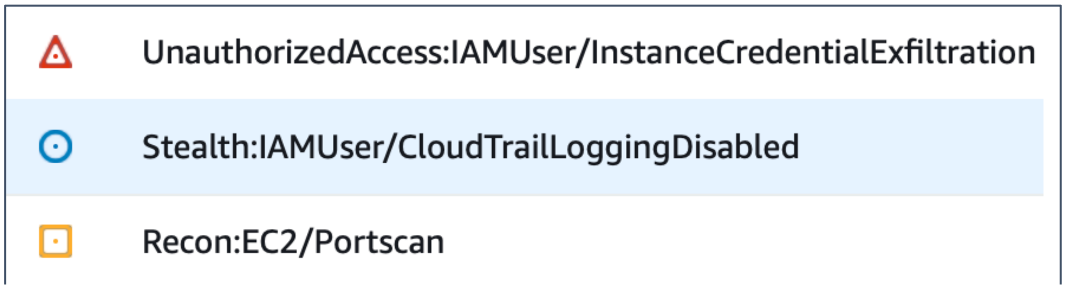
AWS CloudTrail records the first copy of control plane or management API calls automatically and without incurring additional charges. These logs provide clarity around events such as the addition or removal of logging, listing of S3 buckets, or describing IAM roles. The source IP address and the principal are recorded for every request, whether it originated from the web console or the command line. This means that even rudimentary searching in CloudTrail data can uncover the type of activity associated with account abuse without additional cost.

3.4.2. Utility of flow logs

Flow logs are limited in these use cases in that not every attack is going to be immediately visible or evident from network activity. Identity and account hijacking threats in particular could be precipitated by substantial configuration changes to identity and access management, even without producing flow logs. If an access token were to be abused in this manner (whether by an insider or not), an attacker could also take care to disable flow logs or modify security groups to allow a specific subset of traffic, thereby avoiding the creation of suspicious logs. Depending on the configuration, an organization may opt to forego the logging of *accept* logs, which means a lack of *reject* logs may go undetected for some time. If instance profile credentials are pilfered from an instance, remote network traffic associated with illegitimate access of the system could be visible in a flow log. If the connection was irregular enough to attract attention, this distinction might be enough to raise an alert issue before an attacker has a chance to abuse the credential.

3.4.3. Utility of GuardDuty

AWS GuardDuty detected and alerted on account abuse when logging configurations were changed and when instance profile credentials were seen being used outside of the instance profile.


UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration

Stealth:IAMUser/CloudTrailLoggingDisabled

Recon:EC2/Portscan

Figure 11: Sample GuardDuty findings related to IAM and account abuse/insider threats.

3.5. Insecure Interfaces and APIs; Data Breaches

In the final simulation undertaken for this research, application layer attacks are executed against the test environment to represent the types of abuses that lead to discovering and exploiting insecure interfaces and APIs. Automated SQLi through sqlmap was the method of choice. Data breaches can occur from many different sources. Still, for this research, the data breaches threat was included with insecure interfaces and APIs and highlights the depth of risk that insecure interfaces and APIs can represent. To simulate this, OWASPs Mutillidae was installed on an EC2 instance running a current version of Ubuntu with a CloudWatch agent configured to send Apache httpd logs to CloudWatch.

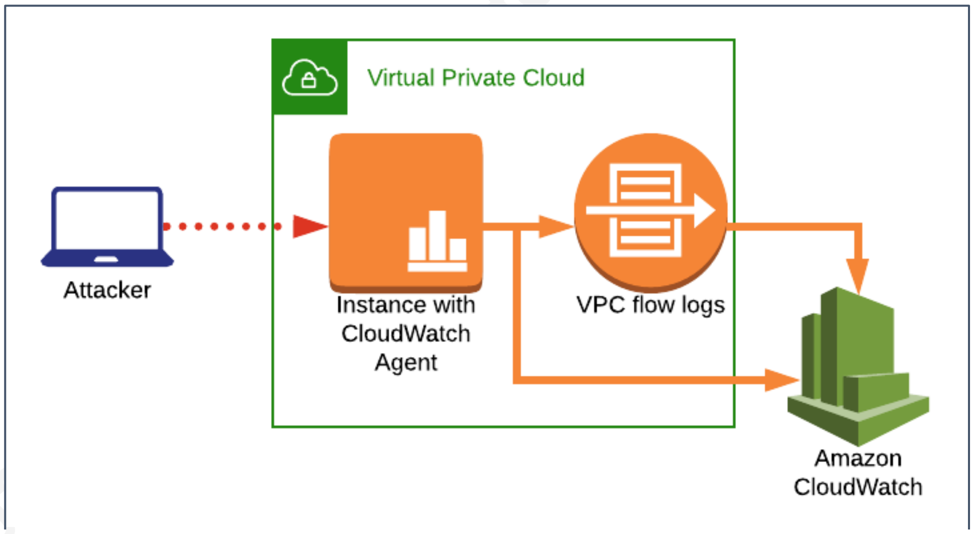
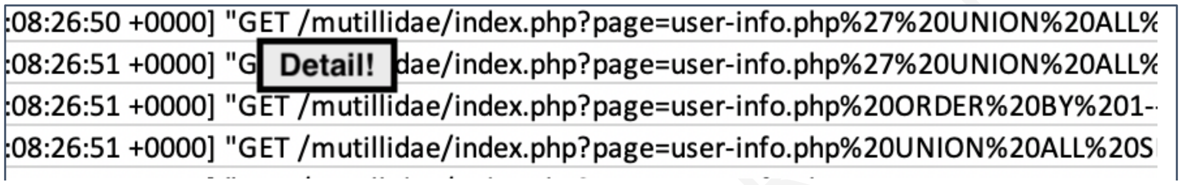


Figure 12: Illustration depicting instance running vulnerable web application (Mutillidae) and also sending apache logs via installed CloudWatch agent and VPC flow logs.

3.5.1. Utility of non-flow logs

Detecting application compromise in the public cloud is perhaps most directly accomplished by collecting application layer logs. Flow logging has no access to the application layer, so it is not limited by encrypted application traffic as, for example, full packet capture can be (Szili, 2019). The caveat here is that shifting from a network-first monitoring approach towards an application-first approach *also* alleviates concerns that application layer network traffic is encrypted in-flight or is otherwise unavailable for further analysis. Simplifying an application-first approach to logging, AWS offers the

CloudWatch agent (Amazon, 2020b), which can be directed to ingest logs from any directory accessible by the agent and send them to CloudWatch.



```
:08:26:50 +0000] "GET /mutillidae/index.php?page=user-info.php%27%20UNION%20ALL%
:08:26:51 +0000] "G  Detail!  dae/index.php?page=user-info.php%27%20UNION%20ALL%
:08:26:51 +0000] "GET /mutillidae/index.php?page=user-info.php%20ORDER%20BY%201-
:08:26:51 +0000] "GET /mutillidae/index.php?page=user-info.php%20UNION%20ALL%20S
```

Figure 13: Application logs sent from the local CloudWatch agent indicating an SQLi attack.

In addition to these logs, CloudWatch metrics can be useful but generally require additional context to prove most useful. This is similar to identifying irregular activity with flow logs, more detailed information would likely be examined. CPU utilization, the number of network bytes, or packets sent and/or received, etc. can be analyzed, as depicted in figure 14.

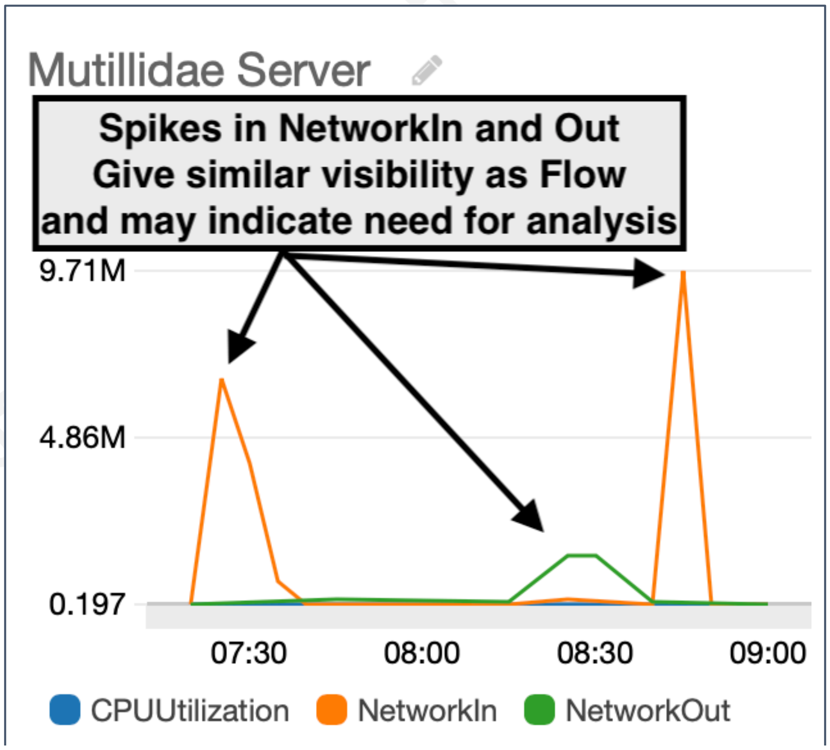
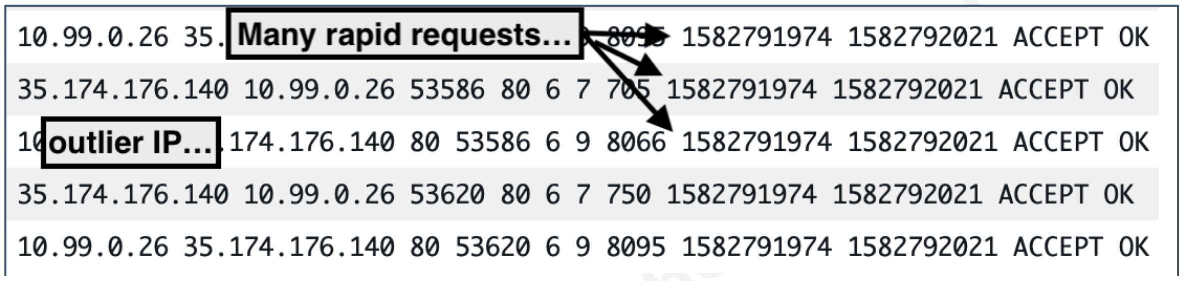


Figure 14: CloudWatch metrics dashboard illustrating spikes in outbound network traffic surrounded by sharp increases in inbound traffic. This is of limited utility without additional context.

3.5.2. Utility of flow logs

As discussed in other sections, flow logs provide limited detail when it comes to application attacks but can provide some insights. Take the below example of flow logs from the time frame the above SQLi attacks were being executed. It is clear a relatively active session existed and that a web server was interacted with, however additional details are just not there, limiting the value flow logs can have in this case.



```
10.99.0.26 35.                          8095 1582791974 1582792021 ACCEPT OK
35.174.176.140 10.99.0.26 53586 80 6 7 705 1582791974 1582792021 ACCEPT OK
10             174.176.140 80 53586 6 9 8066 1582791974 1582792021 ACCEPT OK
35.174.176.140 10.99.0.26 53620 80 6 7 750 1582791974 1582792021 ACCEPT OK
10.99.0.26 35.174.176.140 80 53620 6 9 8095 1582791974 1582792021 ACCEPT OK
```

Figure 15: flow logs during the SQLi attack showing how a single IP becomes the predominant source of activity and how many flows are being created in a single block of time.

An example where flow logs could prove useful, however, is in detecting application layer denial of service attacks. Although DoS attacks were removed from the 2019 *CSA Top Threats* document, application DoS attacks can reasonably be included under insecure interfaces and APIs. DoS attacks were not performed here due to the AWS penetration testing policy prohibiting the testing of denial of service conditions (Amazon, n.d. b). A request to Amazon for a limited exception for research purposes was gracefully denied. The following is an academic analysis of how flow logs could be expected to benefit the detection of application layer DoS attacks.

Two well-documented application layer DoS attacks are Slow Loris and HTTP Floods. Slow Loris seeks to deplete the resources available to the application by trickling partial HTTP requests to the server (OConnor, 2011). HTTP Floods send many web requests in an attempt to overwhelm the application (Radware, 2016, as cited in Freeman, 2017). Both attacks are not visible as attacks from the network or transport layers and cannot be mitigated solely through the extensive bandwidth available to CSP data centers. Flow logging could highlight cases where connections were abnormally long or transferred an unusually low number of bytes or were part of an unusually large number of requests.

Even if it were permissible to simulate DoS conditions in a public cloud environment, this endeavor might have proven profitless, as DoS-focused logging itself may not be very beneficial. This is due to how modern cloud applications can scale to meet demand. Even though a prolonged DoS attack may be costly in extreme cases, both AWS (Amazon, n.d. f) and Azure (Microsoft, 2020) offer DoS cost protection. This protection as a service furnishes a form of pseudo-insurance that empowers customers to recover a portion of the cost incurred to scale in response to a DoS attack. Depending on the business case, and especially for organizations seeking to curtail expenses, building scalability into applications and enabling DoS cost protection services may be sufficient and therefore minimize the value of logging for the purpose of DoS monitoring.

3.5.3. Utility of GuardDuty

This simulation did not trigger alerts in GuardDuty since it has no application layer visibility because its data sources are CloudTrail, DNS, and Flow logs (Amazon, n.d. d). If the hosts devoted to this simulation were engaged for communication to novel or blacklisted systems or addresses outside of the AWS account, that might have triggered alarms. Still, the alert would have been based on unexpected network activity, not application layer behavior, as simulated here.

3.6. Summary Tables

Table 4 below summarizes results by simulation. Table 5 provides additional precision surrounding which non-flow logs provided meaningful observation for each simulation. A green box with a "√"was entered if the solution offered a clear advantage in detection. A gray box with an "X" was recorded if there was no detection available for the simulation. A blue box with a "~" was entered if some detection was possible but experienced limitations for one or more threats. Rows for non-flow and flow are mutually exclusive in this context, so they cannot both be checked under the same column, but GuardDuty is additive and may exist alongside flow or non-flow logs.

	SIM 1	SIM 2	SIM 3	SIM 4	SIM 5
Non-flow	~	X	√	√	√
Flow	√	√	~	~	~
GuardDuty	X	X	~	√	~

Table 4: Summary of simulation detection results.

	SIM 1	SIM 2	SIM 3	SIM 4	SIM 5
CloudTrail (Management, S3 Data)	√	X	√	√	X
CloudWatch (RDS, Custom)	X	X	√	X	√

Table 5: Breakdown of non-flow log types used.

Table 5 shows a relative cost comparison between log types based on observed log sizes and expected volume. This information will aid in the decision to enable logs of a particular class over another. For example, custom application logs are verbose (large) and are likely to generate many logs. A database may only log errors, or only log a small status message in the case of a successful query and not all traffic in a VPC is expected to result in a database query. Flow logs will thus likely be much more frequent and may be logged by both source and destination VPC for traffic between multiple VPCs owned by the same organization.

Both flow and non-flow logs come at a cost. During this limited testing, an average flow log was observed to be approximately 112 bytes. An average RDS log was around 145 bytes. Each transaction in the application logs, by comparison, were on average, 333 bytes. Although almost three times larger than flow logs at first glance, flow logs can be generated on both sides of a transaction for inter-VPC traffic. Logging of both sides of the connection means the actual size cost of flow logs could be closer to 220 bytes or more for a significant portion of network traffic occurring in a VPC, depending on its architecture. In addition to this, the log data provided by those flows are not always independently adequate, often requiring other data sources to provide needed details. Perhaps the most significant consideration when it comes to the cost of logging is ensuring that value is gained from enabling the logs. If regulation or other requirements dictate a network audit trail, then flow logging brings direct benefit for the cost. The environment is all but guaranteed to receive minimal value if there is pressure to implement logging, yet little support to engage that logging for a defined purpose.

	Relative Cost
CloudTrail Management	Included
CloudTrail S3Data	$$
CloudWatch RDS	$
CloudWatch Custom Application	$$$
CloudWatch (Flow)	$$

Table 6: Illustration of estimated relative cost between logging types.

4. Recommendations and Implications

Between CloudTrail Management API calls, logs sent to CloudWatch and services which monitor various log types in an automated fashion, there are several means to monitor for intrusions in a cloud environment. Flow logging provided valuable detail which cannot be obtained from other log sources in simulations one and two. Non-flow logs provided additional detail in simulations three, four, and five. GuardDuty is useful for specific use cases and provided particular value in the credential abuse, account hijacking, and insider threat domain, covering simulation four.

4.1. Final recommendation on to ebb or flow

Enabling flow logging can lead to intriguing findings, especially when seeking to understand unexpected behavior, as exemplified in simulation two. The quandary for flow logging is not dissimilar from that of other logging mechanisms, however. Without proper care and tuning, flow logs (like any other log) can become a cost burden. Adding to the difficulty of the decision to enable flow logs is that cloud environments make alternative logging mechanisms available, which can provide more detailed vistas. It is easy to understand why flow logs could be considered imperative in a cloud environment since they have been a staple of best practice in non-cloud environments, and they seem small enough to be affordable. In large, pay-as-you-go cloud networks, however, this is not always the case, and both collection and storage fees can add up quickly. For a quicker path to value in cloud environments, consider logging application, data plane, and other specific data sources first. The most relevant data requiring the narrowest amount of analysis to understand the activity should be prioritized (Szili, 2019). This research showed that alternatives to flow logging can provide candid visibility into top threats for cloud computing. Unless armed with a specific network monitoring need, this research recommends to ebb rather than flow. Put differently: start with non-flow logs unless requirements exist to track flows.

4.2. Implications for Future Research

A simple place for future research would be to focus on tactical issues such as how spoofing and packet manipulation are handled in other cloud service providers' environments.

Additional research could seek to create estimates for the size of flow logging implementations based on organization size. With a potentially hefty price tag associated with flow logging, having a better understanding of what to expect from small, medium, or large cloud deployments would help security professionals estimate the potential costs more accurately and determine if the associated price tag is worth the expected value.

Individualized research into the top threats facing cloud environments, may provide additional insight into precisely which logging capabilities an organization would be best served to enable for which use case, and why.

5. Conclusion

As organizations migrate to public cloud environments, traditional monitoring methodologies are conceptually uncomplicated to graft in. With numerous options for logging available to help detect and defend against the top cloud threats facing organizations today, knowing what logging to enable is not always instinctual. By grouping cloud threats into manageable groups and running simulations for each of them in a cloud environment, this research showed that there are many cases where non-flow logs were able to detect cloud threats. In many cases, these logs were more direct than flow logs were. GuardDuty and comparable services exhibit promise as a way to quickly enable monitoring for specific needs but are currently limited by what detections are offered and the visibility available to the services. While there are costs associated with almost any form of logging in a cloud environment, those logs that generate demonstrable value in detecting consensus-driven threats are more straightforward to justify than expenditures without such a basis. This research showed that there are log sources that can provide candid visibility into top threats for cloud computing. This research recommends starting with non-flow logs unless specific requirements define an obligation for flow logging, and then implementing flow logging as needed.

References

Amazon. (n.d. a). *VPC Flow Logs.* AWS. https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html

Amazon. (n.d. b). *AWS Customer Support Policy for Penetration Testing.* AWS. https://aws.amazon.com/security/penetration-testing/

Amazon. (n.d. c). *Amazon Cloudwatch pricing.* AWS. https://aws.amazon.com/cloudwatch/pricing/

Amazon. (n.d. d). *Amazon Guard Duty Features.* AWS. https://aws.amazon.com/guardduty/features/

Amazon. (n.d. e). *Vulnerability Reporting.* AWS. https://aws.amazon.com/security/vulnerability-reporting/

Amazon. (n.d. f). *AWS Shield Features.* AWS. https://aws.amazon.com/shield/features/

Amazon. (2016, February 9). *VPC Security Capabilities.* AWS. https://d1.awsstatic.com/aws-answers/VPC_Security_Capabilities.pdf

Amazon. (2017, November 29). *Introducing the Amazon Time Sync Service.* AWS. https://aws.amazon.com/about-aws/whats-new/2017/11/introducing-the-amazon-time-sync-service/

Amazon. (2020a, March 9). *AWS Virtual Private Cloud User Guide.* AWS. https://docs.aws.amazon.com/vpc/latest/userguide/vpc-ug.pdf#flow-logs

Amazon. (2020b, March 9). *Amazon CloudWatch Logs User Guide.* AWS. https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/cwl-ug.pdf#QuickStartEC2Instance

Cloud Security Alliance. (2019, August 6). *The Egregious 11 Cloud Computing Top Threats in 2019.* Cloud Security Alliance. https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-egregious-eleven.

Dickinson, K. (2019, October 30). *How to Perform a Security Investigation in AWS A SANS Whitepaper.* SANS. https://www.sans.org/reading-room/whitepapers/analyst/perform-security-investigation-aws-whitepaper-39230.

Dugan, N. (2019, October 30) *An AWS Network Monitoring Comparison.* SANS. https://www.sans.org/reading-room/whitepapers/cloud/aws-network-monitoring-comparison-39235.

Freeman, M. (2017, January 25). *Building and Maintaining a Denial of Service Defense for Businesses.* SANS. https://www.sans.org/reading-room/whitepapers/infosec/building-maintaining-denial-service-defense-businesses-37552

Google. (n.d. a). *Network pricing.* Google Cloud. https://cloud.google.com/compute/network-pricing#network_telemetry

Google. (n.d. b). *Pricing for Google Cloud's operations suite.* Google Cloud. https://cloud.google.com/stackdriver/pricing

Google. (2020, March 3). *Using VPC Flow Logs.* Google Cloud. https://cloud.google.com/vpc/docs/using-flow-logs

Gul, I. & Hussein, M. (2011, September). Distributed Cloud Intrusion Detection Model. *International Journal of Advanced Science and Technology.* 34, 71-82. https://www.researchgate.net/publication/266291599_Distributed_Cloud_Intrusion_Detection_Model

Hubbard, J., Henderson, J., Valenzuela, I.. (2018, October 24). "Your SIEM Questions Answered." SANS. https://www.sans.org/blog/your-siem-questions-answered/. Retrieved February, 2020.

IANA. (2020, January 31). *Protocol Numbers.* IANA. https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Kumar, M. (2014). Distributed Intrusion Detection System Scalability Enhancement Using Cloud Computing. *GESJ: Computer Science and Telecommunications, 1(41), 16-29.* https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=96249878&site=ehost-live

Li, Z., Xu, H, & Liu, Y. (2017, January 19). A differential game model of intrusion detection system in cloud computing. *International Journal of Distributed Sensor Networks, 13(1), 1-12.* *https://doi.org/10.1177/1550147716687995*

Microsoft. (n.d.). *Network Watcher pricing.* Azure. https://azure.microsoft.com/en-us/pricing/details/network-watcher/

Microsoft. (2017, February 2). *Introduction to flow logging for network security groups.* Azure. https://docs.microsoft.com/en-us/azure/network-watcher/network-watcher-nsg-flow-logging-overview

Microsoft. (2019a, June 6). *What is Azure Security Center?* Azure. https://docs.microsoft.com/en-us/azure/security-center/security-center-intro

Microsoft. (2019b, August 5). *Azure SQL Database Advanced Threat Protection for single or pooled databases.* Azure. https://docs.microsoft.com/en-us/azure/sql-database/sql-database-threat-detection

Microsoft. (2020, January 22). *Azure DDoS Protection Standard overview.* Azure. https://docs.microsoft.com/en-us/azure/virtual-network/ddos-protection-overview

OConnor, T. (2011, December 30). *The Jester Dynamic: A Lesson in Asymmetric Unmanaged Cyber Warfare*. SANS. https://www.sans.org/reading-room/whitepapers/attacking/jester-dynamic-lesson-asymmetric-unmanaged-cyber-warfare-33889

Szili, D. (2019, August). *How to Build a Threat Detection Strategy in Amazon Web Services (AWS).* SANS. https://www.sans.org/reading-room/whitepapers/cloud/paper/39155

Wongthai, W., Rocha, F. L. & Moorsel, A. V. (2013, March). A Generic Logging Template for Infrastructure as a Service Cloud. *International Conference on Advanced Information Networking and Applications Workshops.* https://doi.org/10.1109/WAINA.2013.108

Taggart, D. (2020, April 16). Res5901. GitHub Repository. Retrieved from https://github.com/d3soteric/res5901