



# Global Information Assurance Certification Paper

Copyright SANS Institute  
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

## Interested in learning more?

Check out the list of upcoming events offering  
"Network Monitoring and Threat Detection In-Depth (Security 503)"  
at <http://www.giac.org/registration/gcia>

# An AWS Network Monitoring Comparison

*GIAC (GCIA) Gold Certification*

Author: Nichole Dugan, [ndugan@gmail.com](mailto:ndugan@gmail.com)

Advisor: *David Hoelzer*

Accepted: 10/7/2019

## Abstract

AWS recently released network traffic mirroring in their environment. As this is a relatively new feature, users of the service in the past have used tools such as Security Onion to monitor traffic using a hosted base model of forwarding network traffic to analyze the data. It may not be apparent to an organization which option works best for them, so an analysis should be done of both the traffic mirroring and host based options to determine the benefits and drawbacks of each method. This paper seeks to compare the two types of network monitoring available in the AWS environment, traffic mirroring and host based, and determine which method is more cost-effective, and, through testing, determine which method generates more alerts.

# 1. Introduction

As cloud datacenter offerings become more affordable, more small and medium-sized companies are moving their hardware infrastructure to the cloud. AWS is one of the most popular choices, and it offers many services to make the transition from an on-premise datacenter to one where hardware exists out of the control of the organization. This lack of control has been a concern for some security and networking professionals who traditionally ran network Terminal Access Points (TAPs) to monitor for malicious activity.

This paper discusses two different methods of monitoring infrastructure in the AWS environment, tests the solutions, and compares and contrasts between the methods. Evaluation of the solutions include the cost to run each, as well as ease of use.

## 1.1. Native AWS Monitoring

AWS makes it easy to set up monitoring of a Virtual Private Cloud (VPC) by setting up VPC Flow Log monitoring. This monitoring can either be turned on for the entire VPC or an individual Elastic Network Interface (ENI). The flow log record contains the following fields (AWS, 2019):

- Version
- Account ID
- Interface ID
- Source IP Address
- Destination IP Address
- Source Port
- Destination Port
- Protocol
- Number of Packets Transferred
- Number of Bytes Transferred

- Start Time
- End Time
- Action Taken by ACLs (ACCEPT or REJECT)
- Log Status (OK, NODATA, SKIPDATA)

An example flow log record is shown below (AWS, 2019).

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6
20 4249 1418530010 1418530070 ACCEPT OK
```

While this data can be useful for troubleshooting issues, it does not give the full packet and does not allow analysis in tools such as Suricata or Zeek.

## 1.2. Traffic Mirroring

On June 25<sup>th</sup>, 2019, AWS announced the availability of traffic mirroring for EC2 instances (AWS, 2019). This option resembles using a network TAP in an on-premise datacenter. The traffic mirroring packet format contains the following fields (AWS, 2019):

- Virtual Network ID
- Source IP Address
- Destination IP Address

This solution cannot monitor all network traffic and excludes ARP, DHCP, Instance Metadata Service, NTP, and Windows activation (AWS, 2019). The documentation for AWS Traffic Mirroring also lists several scaling limitations for traffic mirroring on an AWS account; however, these limitations should not be a factor for a small or medium-sized organization.

## 1.3. Security Onion

Security Onion is an open-source Linux distribution that combines many popular monitoring tools in one location. Security Onion focuses on tools that help with intrusion detection, enterprise security monitoring, and log management. It contains the following tools (Security Onion Solutions, 2019):

- Elasticsearch, Logstash, Kibana (ELK)
- Snort
- Suricata
- Zeek, formerly Bro
- Wazuh, formerly OSSEC
- Sguil
- Squert
- CyberChef
- NetworkMiner

This paper focuses primarily on using Wazuh as a Host Intrusion Detection (HID) agent, along with using Zeek and Suricata to monitor traffic from the AWS traffic mirroring option. Currently, Zeek and Suricata are the only options for traffic mirroring in AWS because they support needed VXLAN decapsulation (AWS, 2019).

## 1.4. Comparison between Options

Security Onion offers various tools that offer similar functionalities, such as Wazuh and Logstash. Both offer forwarding of event logs from Windows servers. Wazuh was chosen over Logstash because Wazuh offers file integrity monitoring along with intrusion detection and log forwarding. This combination provides a stronger security base by monitoring system files as well as system events.

## 2. Environment Setup

For testing purposes, the environment has a flat structure. There is one server, a domain controller, with the Wazuh agent installed on it that communicates with the Security Onion server. The Security Onion server is a c5.2xlarge class of server, which has eight virtual CPUs, and 32 GB of memory. The c class of servers are optimized for CPU performance, making them an excellent choice to process traffic data. Because AWS does not easily allow for the creation of an instance from an ISO, it was installed on an Ubuntu 16.04 server. Security Onion was then able to be installed using the apt-get command (Security Onion, 2019), but the AWS Ubuntu image does not have a GUI

Nichole Dugan, ndugan@gmail.com

version, so X Windows forwarding was done with Xming. This installation of Ubuntu 16.04 needed a change to the `/etc/hosts` file to add the private IP address of the server along with the hostname of the server, which takes the format of `ip-192-168-1-0`. In the AWS environment, it is best not to configure the interfaces through the Security Onion setup wizard since this action may cause the instance to become inaccessible.

Another consideration for the setup of Security Onion in an AWS environment is that the security groups must allow traffic on the port from the server or subnet sending in traffic. These additions should occur after running the `so-allow` command in Security Onion. For example, to allow Wazuh to communicate with the Security Onion server, `so-allow` with the "o" option needs to be run, and the security group attached to the instance needs to allow port 1514 through from the monitored server.

While this configuration for Security Onion works for testing purposes, the documentation for a production installation includes at least three servers. Documentation details including a master node to control other nodes, one or more forwarding nodes that can ingest network traffic data, and one or more storage nodes to run Elasticsearch and Logstash. Security Onion's recommended architecture, shown in Figure 1, displays several servers (Security Onion, 2019). This configuration would give a more robust architecture for a production environment and would utilize the most CPU for Elasticsearch. Zeek is also CPU-intensive and requires as much network bandwidth as the data sent to the Zeek sensor.

Security Onion - Distributed Deployment  
Created By Security Onion Solutions

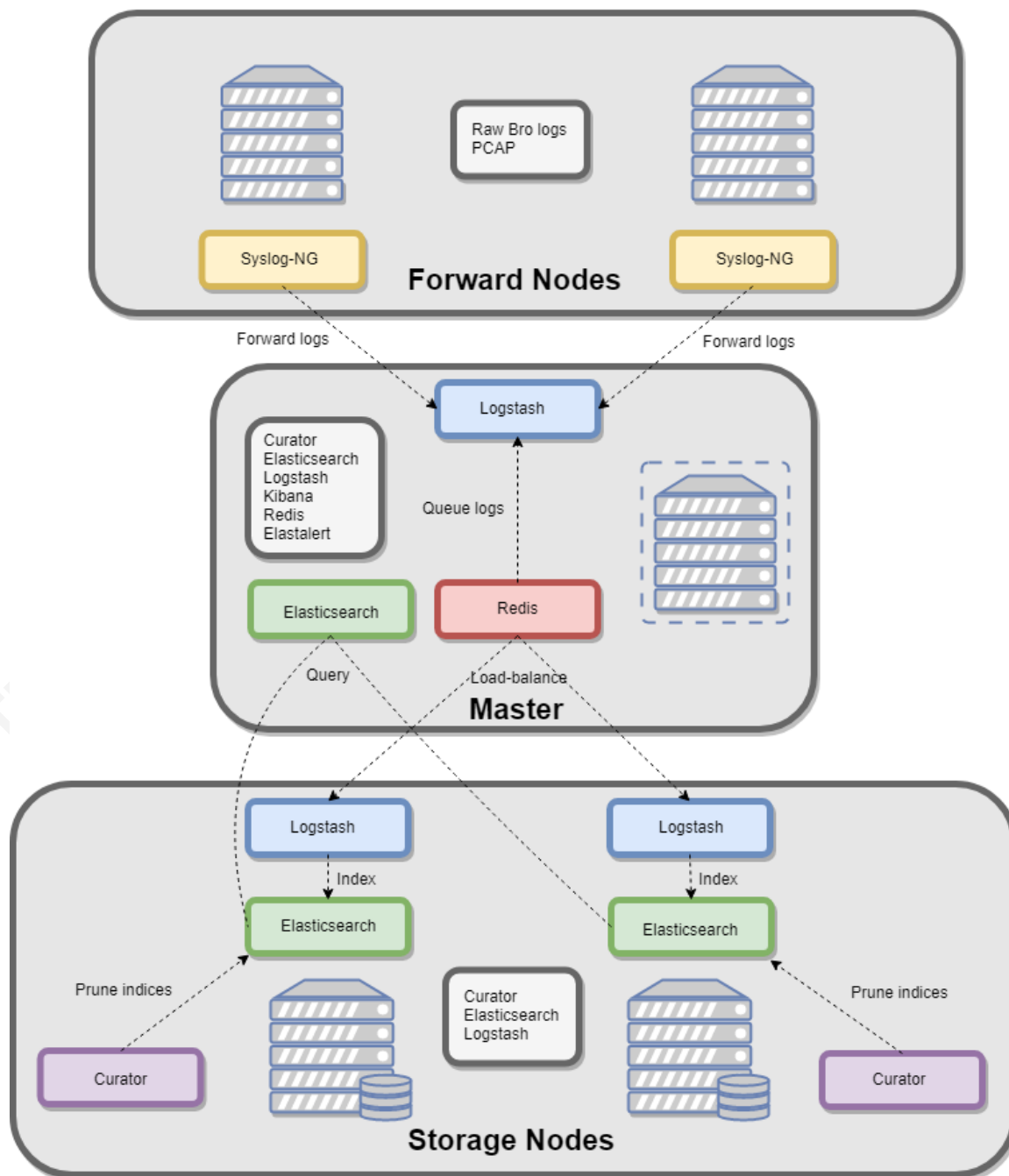


Figure 1: Security Onion Recommended Architecture

## 2.1. AWS Traffic Mirroring

AWS Traffic Mirroring allows for the mirroring of one network interface to another network interface. This mirror can either be from one EC2 instance to another, or from a network interface to a network load balancer. If using a load balancer to distribute the traffic to multiple servers, the load balancer must allow traffic on port 4789, the port for VXLAN traffic.

Understanding this configuration illustrates one difference between network monitoring in AWS versus an on-premise network: we are only able to monitor traffic going to a specific instance. This AWS solution might overwhelm a single sensor, so many sensors may need to be deployed into the network, increasing the cost of the solution. As discussed previously, AWS lists some traffic that cannot be monitored, including ARP and DHCP traffic, which could be used to determine new machines in the environment.

## 2.2. Security Onion

Security Onion is a package of several open-source tools that help with intrusion detection, network monitoring, and log management. Each of the tools can be used separately or in conjunction with each other. Each item provides benefits; for example, Wazuh can detect events that occur on a host, such as software installations or privilege escalation. Zeek and Suricata can monitor network traffic and alert on specific conditions. Security Onion also provides tools for an analyst to investigate activity on the network, including Kibana, Squert, Sguil, and Wireshark.

## 3. Testing the Solution

After Security Onion was installed, Wazuh was installed on a host, and traffic mirroring was set up between the host and the Security Onion server. Tests of the solution were comprised of events that follow the CIS Top 20 Critical Security Controls including Inventory and Control of Hardware Assets, Inventory and Control of Software, Controlled Use of Administrator Privileges, Limitation and Control of Network Ports, Secure Configuration for Devices, and Boundary Defense (Center for Internet Security, 2019).

Nichole Dugan, ndugan@gmail.com



### 3.1. Inventory and Control of Hardware Assets

To test the Inventory and Control of Hardware Assets Critical Security Control in AWS in a virtual environment, a new EC2 instance was created. While Wazuh can create an IAM account to monitor AWS services (Wazuh, 2019), setting up this configuration was out of the scope of this research. Zeek or Suricata may have been able to detect this new server on the network, however the server did not send information to the monitored network interface, so this new server went undetected. The Kibana dashboard in Security Onion could provide a useful starting point by reviewing the “Connections” report listed under the “Bro Hunting” section.

### 3.2. Inventory and Control of Software Assets

The Inventory and Control of Software Assets critical security control was tested by installing Chrome on the monitored server. Wazuh can monitor windows event logs, and the events associated with installing and uninstalling an application are 11707 and 11724, respectively. Reviewing the rules for Wazuh in /var/ossec/rules/0220-msauth\_rules.xml, shows that rule IDs 18146 and 18147, shown below, are monitored and sends an alert by email.

```
<rule id="18147" level="5">
  <if_sid>18101</if_sid>
  <id>^11707$</id>
  <options>alert_by_email</options>
  <description>Windows: Application Installed.</description>
</rule>

<rule id="18146" level="5">
  <if_sid>18101</if_sid>
  <id>^11724$</id>
  <options>alert_by_email</options>
  <description>Windows: Application Uninstalled.</description>
</rule>
```

While the email was not sent, the action was logged in `/var/ossec/logs/alerts/2019/Aug/ossec-alerts-27.log`. The alert is shown below and references the registry key that was involved in the installation event.

```
** Alert 1566937963.309047958: -
ossec,syscheck,pci_dss_11.5,gpg13_4.11,gdpr_II_5.1.f,
2019 Aug 27 20:32:43 ([Server Name]) [Server IP]->syscheck-
registry
Rule: 554 (level 5) -> 'File added to the system.'
File
'HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\EventLog\Ap
plication\Chrome' was added.
```

Attributes:

- MD5: b0a4f0b951abba225dd873dd24883261
- SHA1: 12559e2aadae68d1dc5d158b59125560682d3470

Wazuh allows alert levels from 1 to 16, and level 5 on the alert holds a higher severity than the default of 3 (Wazuh, 2019). Alerts at this level should send an alert via email or forward on to a SIEM for further analysis.

### 3.3. Controlled Use of Administrator Privileges

To help check the usage of administrator privileges as defined in the critical security control Controlled Use of Administrator Privileges, Windows event IDs 577 and 4673 can be monitored as shown in the Wazuh rule listed below.

```
<rule id="18108" level="4">
  <if_sid>18105</if_sid>
  <id>^577$|^4673$</id>
  <description>Windows: Failed attempt to perform a privileged
operation.</description>

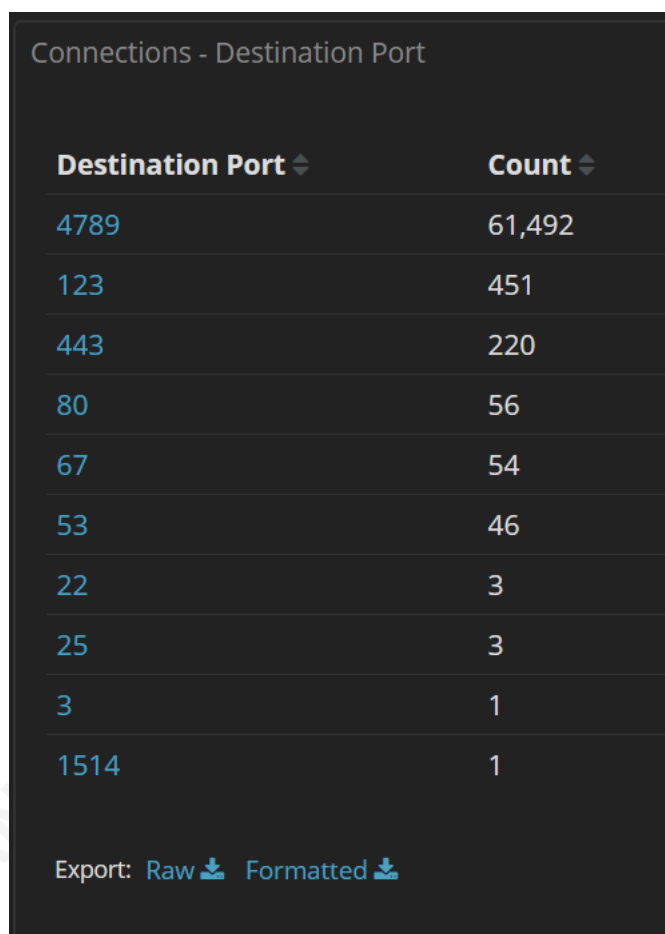
<group>authentication_success,pci_dss_10.2.2,gdpr_IV_32.2,</group>
>
</rule>
```

Nichole Dugan, ndugan@gmail.com

An example of an alert that was fired while testing is shown in Appendix A. The alert gives the user who is attempting the privileged action, the host they are attempting the privileged action on, and the process involved in the alert. These alerts can help a security operations team monitor the network.

### **3.4. Limitation and Control of Network Ports**

To monitor the Limitation and Control of Network Ports critical security control in AWS, additions and changes to security groups should be monitored. While changes to AWS security groups could be added in Wazuh, as mentioned in Section 3.1, reviewing the Kibana dashboard for “Connections” under the “Bro Hunting” section can help monitor open ports between servers in AWS. The version of Zeek included with Security Onion was unable to decode the VXLAN traffic, so the traffic mirroring port between the monitored server and the Security Onion server shows the most traffic as the VXLAN port 4798, as shown in Figure 2.



Destination Port	Count
4789	61,492
123	451
443	220
80	56
67	54
53	46
22	3
25	3
3	1
1514	1



Export: [Raw](#)  [Formatted](#) 

Figure 2: Incoming Port Connection Counts

### 3.5. Secure Configuration for Devices

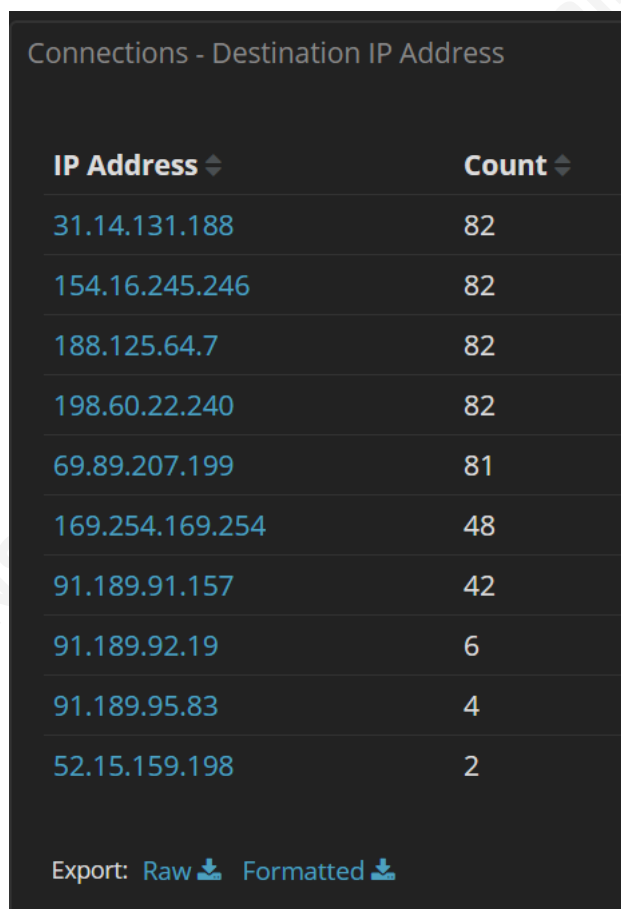
The Secure Configuration for Devices critical security control tests if the default credentials are on a device. By default, when a Windows EC2 instance is created in AWS, the Administrator account has a 30-character password with a mix of lowercase, uppercase, numbers, and special characters.

Wazuh tracks failed login attempts to the system. A sample alert, shown in Appendix B, displays the format of the alert. The alert gives the username, server the user was trying to log into, and the timestamp. These values can help determine if an attack is taking place.

### 3.6. Boundary Defense

The Boundary Defense critical security control monitors connections into and out of the perimeter of the organization's internal network. To review the connections between

the EC2 instances and external IPs, the “Connections” dashboard under the “Bro Hunting” section can help an investigator review external network traffic. An example of the report is shown in Figure 3. This data can be used to compare IP addresses with known malicious IP addresses from threat intelligence sources.



IP Address ↕	Count ↕
31.14.131.188	82
154.16.245.246	82
188.125.64.7	82
198.60.22.240	82
69.89.207.199	81
169.254.169.254	48
91.189.91.157	42
91.189.92.19	6
91.189.95.83	4
52.15.159.198	2



Export: [Raw](#)  [Formatted](#) 

Figure 3: Outgoing IP Address Connections

## 4. Testing Results

During testing, several events were triggered to mimic real-world attacks described by CIS's Top 20 Critical Security Controls. A combination of Zeek alerts and Wazuh alerts were able to detect some of the attempts, but network monitoring in AWS does not provide as much visibility as an on-premise network. A prime weakness of this feature is the inability to monitor the network for new devices.

Zeek provided visibility in control of network ports and boundary defense, but Wazuh provided visibility with inventory and control of software assets, monitoring

privileged accounts, and secure configuration of devices. Both options require configuration changes to allow for proper alerting on these events. As with an on-premise network, a combination of several security controls is needed to minimize security weaknesses.

The most significant problem with this solution was Zeek failing to distinguish between the forwarded traffic and traffic coming into the Security Onion sensor. This problem caused Zeek to interpret the majority of traffic to be on port 4798 and not the ports associated with the original traffic. Further testing was done between the latest version of Zeek and the version of Zeek included in Security Onion 16.04. The results of the testing process are included in section 5.

## 5. VXLAN Testing

The AWS networking environment uses VXLAN tunneling to do traffic mirroring. Zeek Version 3.0.0 released on August 8<sup>th</sup>, 2019 supports decapsulation of VXLAN traffic (Zeek, 2019). The version of Bro included in Security Onion 16.04 is 2.6.3. The latest version of Zeek was installed on a separate Ubuntu server, and further tests were completed.

Two tests were performed on each instance to determine the difference between the Zeek and Bro instances in AWS. The first test attempted to connect to the Windows server using RDP and the incorrect credentials. The second test performed was a ping from a workstation to the Windows server.

The test results showed that the Security Onion sensor was unable to pick up either the ping or RDP traffic from the workstation. The sensor running the latest version of Zeek was able to detect the traffic from the workstation and place the traffic in the proper file, rdp.log or conn.log.

More research needs to be done to determine if the latest version of Security Onion can be upgraded to use the latest version of Zeek or if a separate Zeek sensor could send its logs to Security Onion. Sending the logs to Security Onion would allow a security team to utilize the reports available in the Kibana dashboard while still using the AWS

Traffic Mirroring solution. A possible solution to this problem is to use FileBeats to send the Zeek logs to the Elasticsearch database hosted in Security Onion (Logz.io, 2018).

## 6. Cost Analysis

To set up the recommended Security Onion architecture, as shown in Figure 1, a minimum of three servers should be run. The master node running Elasticsearch should remain at a c5.2xlarge while the forward and storage nodes would each be c5.xlarge. The c5 class of servers emphasize CPU performance, which is required for processing of network traffic. A breakdown of the costs to run this configuration is shown below.

- Master node - \$0.34/hour, \$2,978.40/year
- Forward node - \$0.17/hour, \$1,498.20/year
- Storage node - \$0.17/hour, \$1,498.20/year
- Traffic mirroring - \$0.015/hour, \$131.40/year per server

With an environment of 25 monitored servers, the total cost for traffic mirroring would be \$3,285/year. The total estimate for running this solution for a year is \$9,259.80. In contrast, the Gigamon GigaVue option available from the AWS Marketplace is \$3.20/hour for a total of \$28,032/year (Gigamon, 2019). If the traffic mirroring is sent to an on-premise target, additional data charges would be incurred, as AWS charges for data downloaded from the VPC.

## 7. Conclusion

Moving to a cloud infrastructure can help mitigate security risks by sharing some security responsibility with the cloud hosting provider. However, as more organizations are breached in the cloud, it is crucial that they monitor their cloud resources to ensure they have not been compromised. Before AWS introduced their Traffic Mirroring options on June 25<sup>th</sup> 2019, other solutions that we proposed involved routing all traffic through a security device controlled by the organization's network administrators as proposed by Radichel (Radichel, 2017). While this solution has the possibility of capturing more data than the traffic mirroring allowed in AWS, it presents a single point of failure. The traffic

mirroring solution offered in AWS needs to forward traffic to a listener, and Security Onion has the ability to read network traffic and generate alerts based on certain conditions.

This paper tested two solutions offered by Security Onion to help detect events that fall into the categories listed by CIS's Top 20 Critical Security Controls. Wazuh was able to detect items that take place within the host, for example, failed login attempts. Zeek was able to detect network items through AWS traffic mirroring such as external IPs accessing internal resources. The Security Onion's Zeek sensor could not decapsulate the VXLAN traffic include in the Traffic Mirroring solution offered by AWS. More research should be done to determine if Security Onion could be upgraded or if Zeek could forward its logs to Security Onion. Wazuh had limitations with alerting and distinguishing between abnormal events and regular events. More configuration changes would need to be made to make a useful solution.

While the cost of this solution would be around \$10,000, this is a considerable cost savings compared to a hosted solution such as the GigaVUE solution provided in the AWS Marketplace. This cost difference may be a factor in small or medium-sized organizations that are willing to take more time to configure an open-source solution such as Security Onion.



## References

- AWS. (2019, June 25). *Announcing Amazon VPC Traffic Mirroring for Amazon EC2 Instances*. Retrieved from AWS: <https://aws.amazon.com/about-aws/whats-new/2019/06/announcing-amazon-vpc-traffic-mirroring-for-amazon-ec2-instances/>
- AWS. (2019). *Traffic Mirroring Limits and Considerations*. Retrieved from AWS Documentation: <https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-considerations.html>
- AWS. (2019). *Traffic Mirroring Packet Format*. Retrieved from AWS Documentation: <https://docs.aws.amazon.com/vpc/latest/mirroring/traffic-mirroring-sessions.html>
- AWS. (2019). *VPC Flow Logs*. Retrieved from AWS Documentation: <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html>
- AWS. (2019). *Working with Open-Source Tools for Traffic Mirroring*. Retrieved from AWS Documentation: <https://docs.aws.amazon.com/vpc/latest/mirroring/tm-example-open-source.html>
- BL King Consulting. (2019). *Security Onion*. Retrieved from AWS Marketplace: [https://aws.amazon.com/marketplace/pp/B07NWLLZ73?qid=1561124889839&sr=0-1&ref\\_=srh\\_res\\_product\\_title](https://aws.amazon.com/marketplace/pp/B07NWLLZ73?qid=1561124889839&sr=0-1&ref_=srh_res_product_title)
- Center for Internet Security. (2019). *CIS Center for Internet Security*. Retrieved from The 20 CIS Controls & Resources: <https://www.cisecurity.org/controls/cis-controls-list/>
- Chuvakin, A. (2010, August 6). *Dr Anton Chuvakin Blog PERSONAL Blog*. Retrieved from Blogspot: [http://chuvakin.blogspot.com/2010/08/updated-with-community-feedback-sans\\_06.html?m=1](http://chuvakin.blogspot.com/2010/08/updated-with-community-feedback-sans_06.html?m=1)
- Gigamon. (2019, August 28). *AWS Marketplace*. Retrieved from GigaVUE Cloud Suite for AWS 5.6 – 100 pack: [https://aws.amazon.com/marketplace/pp/prodview-ptts57jwmr2mu?qid=1567018756766&sr=0-1&ref\\_=srh\\_res\\_product\\_title](https://aws.amazon.com/marketplace/pp/prodview-ptts57jwmr2mu?qid=1567018756766&sr=0-1&ref_=srh_res_product_title)
- Logz.io. (2018, March 12). *Integrating Bro IDS with the ELK Stack – Part 1*. Retrieved from Logz.io: <https://logz.io/blog/bro-elk-part-1/>
- Radichel, T. (2017, August 10). *Packet Capture in AWS*. Retrieved from SANS Reading Room: <https://www.sans.org/reading-room/whitepapers/cloud/packet-capture-aws-37905>
- Security Onion. (2019). *Architecture*. Retrieved from Security Onion Documentation: <https://securityonion.readthedocs.io/en/latest/architecture.html#distributed>
- Security Onion. (2019). *Production Deployment*. Retrieved from Security Onion Documentation: <https://securityonion.readthedocs.io/en/latest/production-deployment.html>
- Security Onion Solutions. (2019). *About*. Retrieved from Security Onion: <https://securityonion.readthedocs.io/en/latest/about.html>

Wazuh. (2019). *Alerts*. Retrieved from Wazuh Docs:

<https://documentation.wazuh.com/3.x/user-manual/reference/ossec-conf/alerts.html#reference-ossec-alerts>

Wazuh. (2019). *Monitoring AWS Services*. Retrieved from Wazuh Docs:

<https://documentation.wazuh.com/3.x/amazon/services/index.html>

Zeek. (2019, August 8). *Zeek Blog*. Retrieved from Zeek: <https://blog.zeek.org/>

.

## Appendix A

```

** Alert 1566913314.257333854: - windows,
windows_security,authentication_success,pci_dss_10.2.2,gdpr_IV_32
.2,
2019 Aug 27 13:41:54 ([Server Name]) [Server IP]->EventChannel
Rule: 60107 (level 4) -> 'Failed attempt to perform a privileged
operation'
{"win":{"system":{"providerName":"Microsoft-Windows-Security-
Auditing","providerGuid":{"54849625-5478-4994-A5BA-
3E3B0328C30D"},"eventID":"4673","version":"0","level":"0","task":
"13056","opcode":"0","keywords":"0x8010000000000000","systemTime"
:"2019-08-
27T13:41:53.433519000Z","eventRecordID":"32648375","processID":"4
","threadID":"5928","channel":"Security","computer":["Server
FQDN"],"severityValue":"AUDIT_FAILURE","message":"A privileged
service was
called."},"eventdata":{"subjectUserSid":["SID"],"subjectUserName"
:["Username"],"subjectDomainName":["Domain"],"subjectLogonId":"0x
456187f","objectServer":"Security","privilegeList":"SeTcbPrivileg
e","processId":"0x1568","processName":"C:\\Windows\\explorer.exe"
}}}}
win.system.providerName: Microsoft-Windows-Security-Auditing
win.system.providerGuid: {54849625-5478-4994-A5BA-3E3B0328C30D}
win.system.eventID: 4673
win.system.version: 0
win.system.level: 0
win.system.task: 13056
win.system.opcode: 0
win.system.keywords: 0x8010000000000000
win.system.systemTime: 2019-08-27T13:41:53.433519000Z
win.system.eventRecordID: 32648375
win.system.processID: 4
win.system.threadID: 5928
win.system.channel: Security

```

Nichole Dugan, ndugan@gmail.com

```
win.system.computer: [Server Name]
win.system.severityValue: AUDIT_FAILURE
win.system.message: A privileged service was called.
win.eventdata.subjectUserSid: S-1-5-21-313415027-1372798643-
1237804090-22119
win.eventdata.subjectUserName: [Username]
win.eventdata.subjectDomainName: [Domain]
win.eventdata.subjectLogonId: 0x456187f
win.eventdata.objectServer: Security
win.eventdata.privilegeList: SeTcbPrivilege
win.eventdata.processId: 0x1568
win.eventdata.processName: C:\Windows\explorer.exe
```

## Appendix B

```

** Alert 1566950401.0: - windows,
windows_security,win_authentication_failed,pci_dss_10.2.4,pci_dss
_10.2.5,gpg13_7.1,gdpr_IV_35.7.d,gdpr_IV_32.2,
2019 Aug 28 00:00:01 ([Server Name]) [IP]->EventChannel
Rule: 60131 (level 5) -> 'Windows DC Logon Failure'
{"win":{"system":{"providerName":"Microsoft-Windows-Security-
Auditing","providerGuid":"{[GUID]}","eventID":"4769","version":"0
","level":"0","task":"14337","opcode":"0","keywords":"0x801000000
0000000","systemTime":"2019-08-
28T00:00:01.302225400Z","eventRecordID":"32699434","processID":"6
08","threadID":"1752","channel":"Security","computer":"[Server
FQDN]","severityValue":"AUDIT_FAILURE","message":"A Kerberos
service ticket was
requested."},"eventdata":{"targetUserName":"[Username]","targetDo
mainName":"[Domain]","serviceName":"[Username]","serviceSid":"S-
1-0-
0","ticketOptions":"0x40810000","ticketEncryptionType":"0xffffffff
f","ipAddress":"::ffff:[IP]","ipPort":"63302","status":"0x1b","lo
gonGuid":"{00000000-0000-0000-0000-000000000000}"}}}
win.system.providerName: Microsoft-Windows-Security-Auditing
win.system.providerGuid: {54849625-5478-4994-A5BA-3E3B0328C30D}
win.system.eventID: 4769
win.system.version: 0
win.system.level: 0
win.system.task: 14337
win.system.opcode: 0
win.system.keywords: 0x8010000000000000
win.system.systemTime: 2019-08-28T00:00:01.302225400Z
win.system.eventRecordID: 32699434
win.system.processID: 608
win.system.threadID: 1752
win.system.channel: Security
win.system.computer: [Server FQDN]

```

Nichole Dugan, ndugan@gmail.com

```
win.system.severityValue: AUDIT_FAILURE
win.system.message: A Kerberos service ticket was requested.
win.eventdata.targetUserName: [Username]
win.eventdata.targetDomainName: [Domain]
win.eventdata.serviceName: [Username]
win.eventdata.serviceSid: S-1-0-0
win.eventdata.ticketOptions: 0x40810000
win.eventdata.ticketEncryptionType: 0xffffffff
win.eventdata.ipAddress: ::ffff:[IP]
win.eventdata.ipPort: 63302
win.eventdata.status: 0x1b
win.eventdata.logonGuid: {00000000-0000-0000-0000-000000000000}
```