# Implementing a Network Intrusion Detection System in a Cloud Environment

Chri Yu Gage Louis Roberto

*Abstract*—**The project goal is to utilize the Amazon Web Services (AWS) platform to create an environment with a Network Intrusion Detection System (NIDS), which provides security intelligence across selected instances using open source technologies. We want to construct an environment to simulate methods of protecting cloud assets against common cyber attacks we see today. We set up a Virtual Private Cloud within AWS that is separated into two subnets, a public and private subnet. The routing rules that we created for our environment allow for the internet to connect to instances in the public subnet, while the private subnet remains only visible to instances within the network. For our Network Intrusion Detection System (NIDS), we are utilizing an open source tool called Snort to monitor traffic and respond to specific network behaviors. The instance that Snort will be installed on will also act as a gateway for traffic to be monitored between the VPC and the Internet, allowing it to identify and shut down threats as if it was deployed in a physical network. Activity detected by Snort is then sent to our chosen Security Information and Event Management (SIEM) tool, Splunk, for further action.**

## I. INTRODUCTION

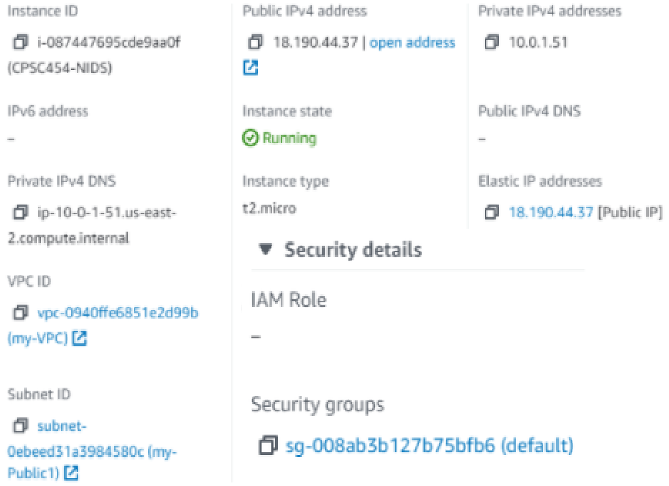
According to a 2008 poll by the IDC Enterprise Panel, security was the top concern of companies that were still reluctant to adopt cloud computing as part of their business model. A 2014 survey of cloud computer security concerns by Issa M. Khalil, Abdallah Khreishah, and Muhammad Azeem concluded that network security issues were the biggest security challenge for cloud computing, because cloud computing depends much more on access to and communication over networks than more traditional types of computing [1]. Denial of service (DoS) attacks alone can render a cloud computing service useless to its clients. Other major network security concerns include man-in-the-middle (MITM) and theft of service (ToS) attacks. As a result, network intrusion detection systems (NIDS) are vital to the security and reliability of cloud computing services. The goal of this project is to create a cloud-based NIDS application. The application is hosted on AWS and receives forwarded network packets to monitor for malicious activity or policy violations. We use a security information and event management (SIEM) tool to parse the data and display any malicious activity on the network. Section II of this paper explains how the network is implemented via VPC and private and public subnets. Section III discusses the implementation of the NIDS, while section IV discusses simulation of intrusion and attacks.

## II. IMPLEMENTING NETWORK

To implement the network we are using Amazon Web Services (AWS). First, we set up a virtual private cloud (VPC) which allows us to have complete control of our virtual networking environment. With a VPC, we are able to control who can access resources on the instances in our environment. By segregating the network into a public and private subnet, we can safely put resources that we want the public to access while also isolating any sensitive resources in the private part of our network [2].

### *A. Setting Up the Virtual Private Cloud*

To set up our VPC on AWS, we first selected the option to create a new VPC and assigned this new VPC a non-routable private IPv4 range of 10.0.0.0/16. Next, we created a new internet gateway and attached it to the VPC. The internet gateway is used by the public subnet of the VPC to allow for communication with the rest of the internet. We then created the public and private subnets within the VPC. We assigned the private IPv4 range 10.0.1.0/24 to the public subnet and the private IPv4 range 10.0.2.0/24 to the private subnet. To connect the public subnet to the internet gateway, Fig. 1. We created a route table that has the internet gateway as a target and 0.0.0.0/0 as a destination, then associated this route table with the



Instance ID
i-087447695cde9aa0f (CPSC454-NIDS)

Public IPv4 address
18.190.44.37 | open address

Private IPv4 addresses
10.0.1.51

IPv6 address
–

Instance state
Running

Public IPv4 DNS
–

Private IPv4 DNS
ip-10-0-1-51.us-east-2.compute.internal

Instance type
t2.micro

Elastic IP addresses
18.190.44.37 [Public IP]

VPC ID
vpc-0940ffe6851e2d99b (my-VPC)

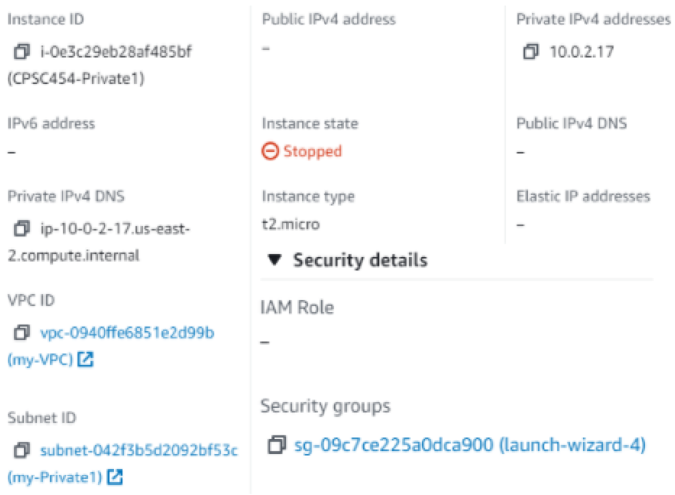
▼ Security details

IAM Role
–

Subnet ID
subnet-0ebeed31a3984580c (my-Public1)

Security groups
sg-008ab3b127b75bfb6 (default)

Fig. 1. *The public subnet EC2 instance.*



Instance ID
i-0e3c29eb28af485bf (CPSC454-Private1)

Public IPv4 address
–

Private IPv4 addresses
10.0.2.17

IPv6 address
–

Instance state
Stopped

Public IPv4 DNS
–

Private IPv4 DNS
ip-10-0-2-17.us-east-2.compute.internal

Instance type
t2.micro

Elastic IP addresses
–

VPC ID
vpc-0940ffe6851e2d99b (my-VPC)

IAM Role
–

Subnet ID
subnet-042f3b5d2092bf53c (my-Private1)

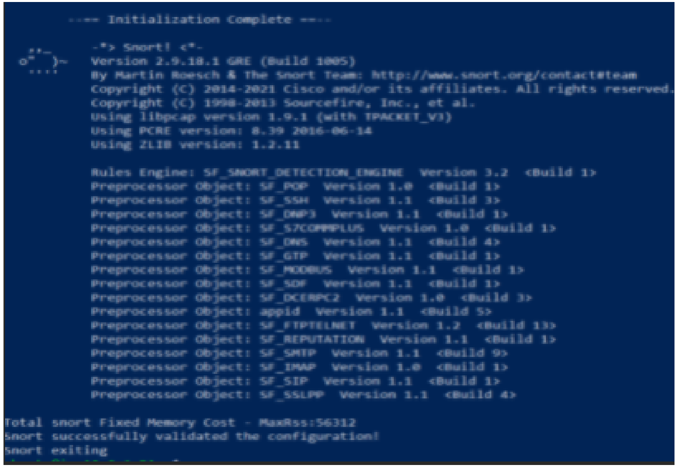
Security groups
sg-09c7ce225a0dca900 (launch-wizard-4)

Fig. 2. *The private subnet EC2 instance.*

public subnet. We also created another route table without the internet gateway as a target, and associated this table with the private subnet. We then created security groups that specified permissions for traffic inbound to and outbound from the EC2 instances that we planned to launch on the public and private subnets. Once the security groups were created, we created EC2 instances for the public and private subnets by launching EC2 instances associated with those subnets and their corresponding security groups. Fig. 1 shows the EC2 instance for the public subnet, and Fig. 2 shows the EC2 instance for the private subnet.

### *B. Installing and Configuring Snort*

Snort is a popular network intrusion detection system (NIDS) that monitors data sent and received through a specific network interface. When configured appropriately, Snort can identify malware, compromised systems, and network policy violations [3]. We installed Snort on the EC2 instance for the public subnet of the VPC by first creating a temp directory where we downloaded and installed the Data Acquisition library (DAQ) used by Snort, and then downloading and installing Snort itself.



```
        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.18.1 GRE (Build 1005)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2021 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.9.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>

Total snort Fixed Memory Cost - MaxRss:56312
Snort successfully validated the configuration!
Snort exiting
```

Fig. 3. *Snort configuration successfully validated.*

To configure Snort, we first modified the snort.conf file. We then validated by testing our configuration as shown in Fig. 3. Once we confirmed that Snort was running, we reconfigured Snort to run in the background so we would not have to manually run it. With the service defined, we reload the systemctl daemon, then start Snort and verify that it is running.

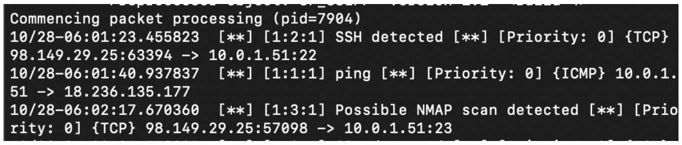
## III. CREATING/TESTING SNORT RULES

In our AWS environment, Snort's role is to monitor traffic coming in through the internet gateway on the public subnet, to ensure the traffic is safe before it continues onto other parts of our network. Snort does this by comparing network activity against a series of rules that help define malicious network activity, and generating alerts whenever detected activity matches a rule [4]. The format for Snort rules is as follows:

```
<action> <protocol> <source ip> <source port> ->
<dest. ip> <dest. port> (rule options)
```

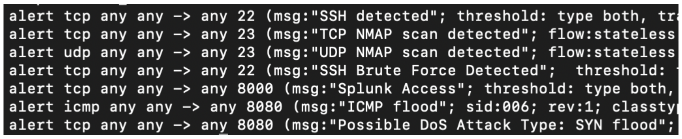
These rules can be configured in the local.rules file located in the Snort installation files. Once Snort is started on our gateway Linux instance, it will create logs and send alerts when incoming network packets match the rules that we configure.

The initial three rules we used to test Snort's traffic tracking in our environment defined malicious behavior in terms of ssh attempts and Nmap scan attempts. Our first rule generates the message "SSH detected" when at least 5 ssh attempts on port 22 are detected within a 30-second interval [6]. The second rule generates the message "Possible NMAP scan detected" after 1 Nmap scan attempt is detected within a 30-second interval. The third rule generates the message "SSH Brute Force Detected" when at least 30 ssh attempts are detected within a 60-second interval. Activities that trigger these three rules are tracked by source. We ran commands from remote computers to trigger these rules in the same way a threat actor would. Fig. 4 shows our Snort results when someone pings, does a Nmap scan on, or attempts to ssh into instances of our environment.



```
Commencing packet processing (pid=7904)
10/28-06:01:23.455823  [**] [1:2:1] SSH detected [**] [Priority: 0] {TCP}
98.149.29.25:63394 -> 10.0.1.51:22
10/28-06:01:40.937837  [**] [1:1:1] ping [**] [Priority: 0] {ICMP} 10.0.1.
51 -> 18.236.135.177
10/28-06:02:17.670360  [**] [1:3:1] Possible NMAP scan detected [**] [Prio
rity: 0] {TCP} 98.149.29.25:57098 -> 10.0.1.51:23
```

Fig. 4: *Snort alerts when someone pings, nmap, and/or ssh into our network.*



```
alert tcp any any -> any 22 (msg:"SSH detected"; threshold: type both, tra
alert tcp any any -> any 23 (msg:"TCP NMAP scan detected"; flow:stateless
alert udp any any -> any 23 (msg:"UDP NMAP scan detected"; flow:stateless
alert tcp any any -> any 22 (msg:"SSH Brute Force Detected";  threshold: 
alert tcp any any -> any 8000 (msg:"Splunk Access"; threshold: type both,
alert icmp any any -> any 8080 (msg:"ICMP flood"; sid:006; rev:1; classty
alert tcp any any -> any 8080 (msg:"Possible DoS Attack Type: SYN flood";
```

*Fig. 5.  Expanded Snort alerts for SSH, NMap, SSH brute force, Splunk access, ICMP flood, and DoS attack*

After further research, we added another three rules to our Snort implementation. The fourth rule generates the message "Splunk access" when there is a detected connection to our Splunk instance (see following section) within a 60-second interval. The fifth rule generates the message "ICMP flood" when at least 500 attempts are detected within 3 seconds. The sixth rule generates the message “Possible DoS Attack TYPE: SYN flood” after 20 attempts are detected within a 10-second interval. Activities that trigger the fifth and sixth rules are tracked by destination.

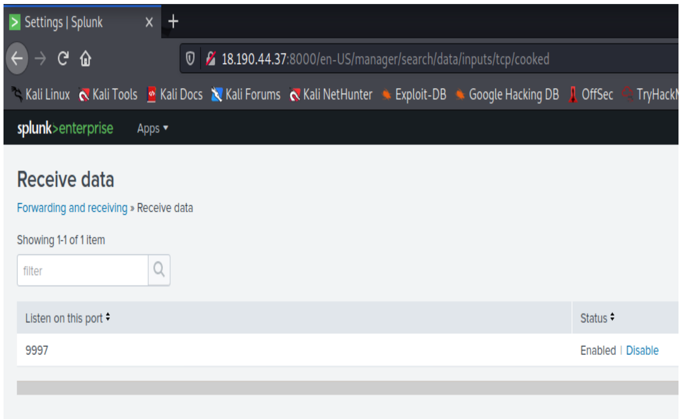
## IV. SIEM

SIEM (**S**imulation, **I**nformation, and **E**vent **M**anagement) is a security method that recognizes potential security threats generated by applications and networks. For our project, we use the software Splunk to implement SIEM. Splunk is a software platform to search, analyze and visualize the machine-generated data gathered from the websites, applications, sensors, devices etc. which make up your IT infrastructure and business. Specifically, Splunk can help analyze and detect potential threats on our cloud network, investigate them across multi-cloud environments, and generate cloud-based security analytics.

The main reason we chose to implement Splunk in our project is because the features it includes even with the community edition are very robust. After installing Splunk, we entered the settings and configured Splunk for receiving data. Fig. 6 shows the Splunk settings and that Splunk is listening and expecting to receive data on port 9997 [5]. Among Splunk’s many features are apps that assist in implementing data aggregation and visualization. Fig. 6 shows some of the apps we used with our Splunk instance, including Search & Reporting, Splunk Essentials for Cloud and Enterprise 8.2, and Splunk Secure Gateway.

The first app, Search & Reporting, helps search, filter and analyze indexed data. The Splunk Search Processing Language (SPL) is used for searching for data from within Splunk after logging in to our Splunk instance. The second app, Splunk Essentials for Cloud and Enterprise 8.2, helps with insights, admin productivity, data infrastructure, and performance. The third app, Splunk Secure Gateway, connects to our Splunk instance to authorize devices via the cloud hosting service Spacebridge, which acts like a secure tunnel for our data. The Splunk Secure Gateway and Spacebridge are built using the Libsodium Encryption Library and TLS 1.2 Encryption protocol, which are used to encrypt data end-to-end so that the data can only be seen by the Splunk instance and the mobile device as shown in Fig. 8.

Next, we implemented the universal forwarder that sends the data from our Snort instance to the Splunk instance. This allows us to create unique dashboards on Splunk using the Snort log data. The universal forwarder looks at logs in var/log/snort and transmits them to the Splunk instance. With the forwarder implemented, we capture new data in our Snort instance and send this data to our Splunk instance.



Receive data

Forwarding and receiving » Receive data

Showing 1-1 of 1 item

| Listen on this port ⬍ | Status ⬍ |
|---|---|
| 9997 | Enabled \| Disable |

*Fig. 6. Forwarding and receiving settings shows that Splunk is listening on port 9997*

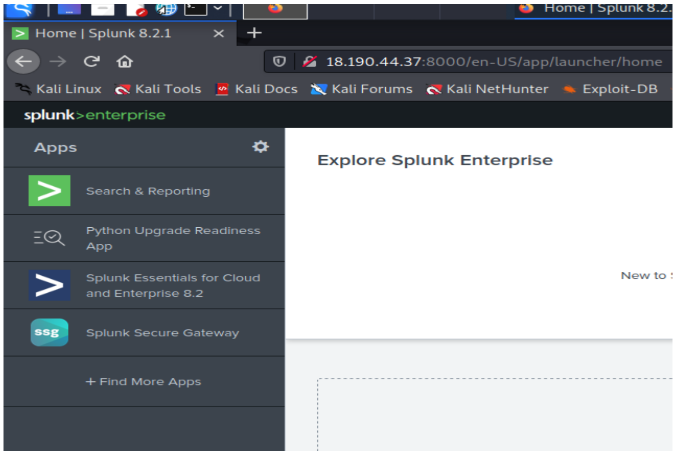


*Fig. 7. App section of Splunk*

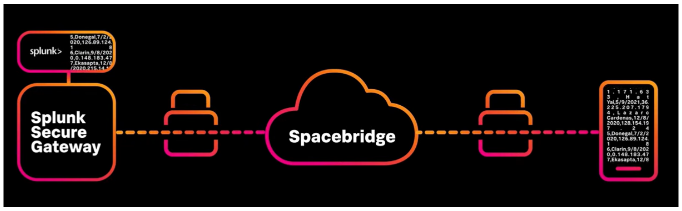
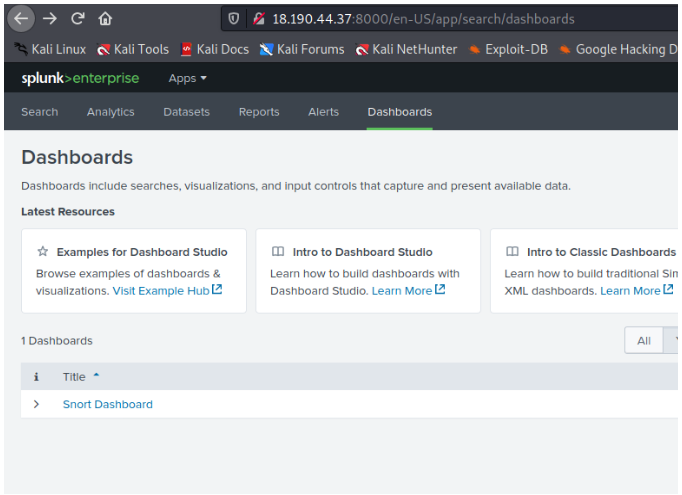


*Fig. 8. Secure Gateway and Spacebridge in action*

The Dashboard section allows us to see the data in many different views and with many filter options. Fig. 9 shows the available dashboards for our Splunk instance, in this case the dashboard created for our project. Fig. 10 shows our dashboard’s pie chart visualization for daily events. Fig. 11 shows the same data as a list and includes a count of the events for that day. Fig. 12 shows two different sections of our dashboard: The left section displays alerts within 5 minutes indicating the time stamp, action (e.g., ssh, nmap), and the source IP. The right section shows the IP addresses of the top 10 activity sources.



Dashboards

Dashboards include searches, visualizations, and input controls that capture and present available data.

Latest Resources

☆ Examples for Dashboard Studio
Browse examples of dashboards & visualizations. Visit Example Hub

Intro to Dashboard Studio
Learn how to build dashboards with Dashboard Studio. Learn More

Intro to Classic Dashboards
Learn how to build traditional Sin XML dashboards. Learn More

1 Dashboards

| i | Title ▴ |
|---|---|
| › | Snort Dashboard |

*Fig. 9. Shows the Dashboard we created "Snort Dashboard"*

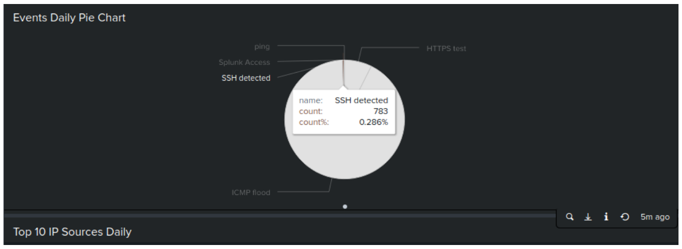
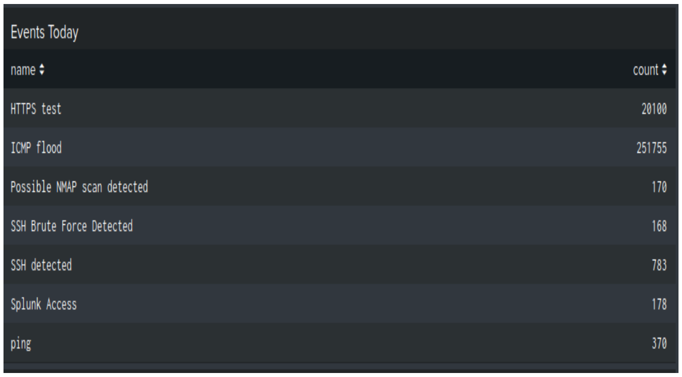


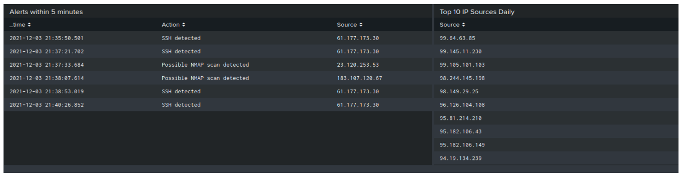
*Fig. 10. Part of Splunk dashboard showing Daily events pie chart*



Events Today

| name ⇕ | count ⇕ |
|---|---|
| HTTPS test | 20100 |
| ICMP flood | 251755 |
| Possible NMAP scan detected | 170 |
| SSH Brute Force Detected | 168 |
| SSH detected | 783 |
| Splunk Access | 178 |
| ping | 370 |

*Fig. 11. Part of Splunk dashboard showing Named Events count*



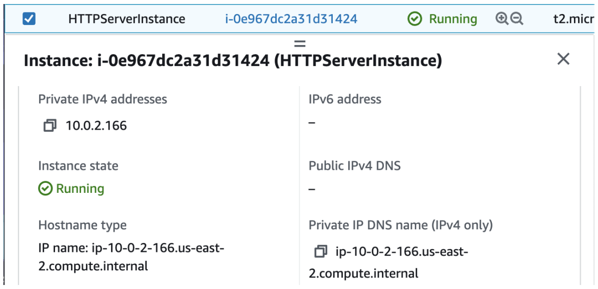
Alerts within 5 minutes

| _time ⇕ | Action ⇕ | Source ⇕ |
|---|---|---|
| 2021-12-03 21:35:50.501 | SSH detected | 61.177.173.30 |
| 2021-12-03 21:37:21.702 | SSH detected | 61.177.173.30 |
| 2021-12-03 21:37:33.684 | Possible NMAP scan detected | 23.120.253.53 |
| 2021-12-03 21:38:07.614 | Possible NMAP scan detected | 183.107.120.67 |
| 2021-12-03 21:38:53.019 | SSH detected | 61.177.173.30 |
| 2021-12-03 21:40:26.852 | SSH detected | 61.177.173.30 |

Top 10 IP Sources Daily

| Source ⇕ |
|---|
| 99.64.63.85 |
| 99.145.11.230 |
| 99.105.101.103 |
| 98.244.145.198 |
| 98.149.29.25 |
| 96.126.104.108 |
| 95.81.214.210 |
| 95.182.106.43 |
| 95.182.106.149 |
| 94.19.134.239 |

*Fig. 12. Part of Splunk dashboard showing Alerts within 5 mins (left) and top 10 IP daily sources (right)*



Instance: i-0e967dc2a31d31424 (HTTPServerInstance)

Private IPv4 addresses: 10.0.2.166

IPv6 address: –

Instance state: Running

Public IPv4 DNS: –

Hostname type: IP name: ip-10-0-2-166.us-east-2.compute.internal

Private IP DNS name (IPv4 only): ip-10-0-2-166.us-east-2.compute.internal

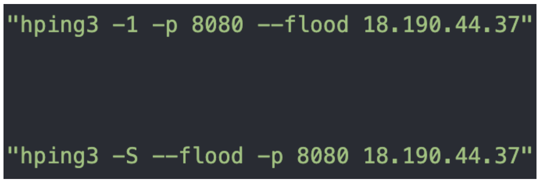
*Fig. 13. EC2 Instance running a HTTP server*

## V. Intrusion Simulation

To ensure that our NIDS is detecting possible malicious activity on our cloud network, we made scripts that can simulate several different types of malicious attacks, such as SSH brute force attacks, DDoS attacks, and Nmap scans. We were able to do this by utilizing various network tools to simulate each respective attack.

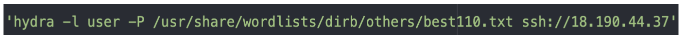
One of the many types of attacks we wanted to take into account when making a network is a DDoS attack. DDoS (Distributed Denial of Service) is considered one of the most common yet effective types of malicious attacks, wherein a network is flooded by an enormous amount of traffic until the network crashes. We simulated this type of attack with a network tool called hping3, which allows users to send various amounts of ICMP and TCP packets to a server or network [7]. For this setup, we prepared a HTTPserver instance (Fig. 13) on AWS that is accessible on port 8080 to the internet that acts as our point of access for a DDoS attack.

The two types of DDoS attacks we are testing in our environment are ICMP and SYN Flood attacks. ICMP Flood attack, also known as a Ping Flood attack, is where an attacker attempts to overwhelm a targeted device with ICMP echo-requests (pings). The result of all of these responses, given by the amount of requests, is a denial of service attack that will make the host unreachable. A SYN Flood attack sends half-open connections, which are initial SYN packets, without completing the handshake. These attacks are used to target individual access points, and are mostly known for attacking firewalls. Because the SYN packets are incomplete, firewalls do not treat them as actual connections and the packets eventually overwhelm the access point or firewall [8]. Fig. 14 shows the hping3 options to simulate such attacks.



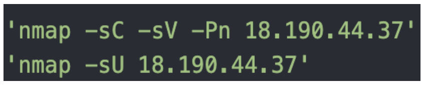
```
"hping3 -1 -p 8080 --flood 18.190.44.37"


"hping3 -S --flood -p 8080 18.190.44.37"
```

*Figure. 14. hping3 commands with flag options for ICMP & SYN flood attacks*



```
'hydra -l user -P /usr/share/wordlists/dirb/others/best110.txt ssh://18.190.44.37'
```

*Figure. 15. Hydra command with user list, password list, and SSH flags.*



```
'nmap -sC -sV -Pn 18.190.44.37'
'nmap -sU 18.190.44.37'
```

*Figure. 16. nmap commands for TCP and UDP*

SSH brute force attacks are another prominent network security concern. Brute force attacks are a trial and error method for attackers attempting to obtain authenticated access to a server. To simulate this type of attack, we used a network tool called Hydra, which is a parallelized login cracker that works with almost any protocol, especially SSH [9]. Since brute force attacks are unpredictable by nature, our brute force Snort rule can ensure that our NIDS will be able to detect a large amount of SSH requests in a given time interval.

To simulate threat actors scanning our network, we utilized the Nmap tool to test our Nmap scan Snort rules. To do this, we used a Nmap command with default script and version flags to attempt a TCP scan of our network. The second Nmap command uses the UDP flag, which sends UDP packets to our network. These Nmap commands are shown in Fig. 16.

## VI. CONCLUSION

By using Snort, Splunk, and other various open-source tools, our group was able to create and implement a NIDS in our AWS cloud environment that will identify threats and monitor traffic to and from our network. Our Snort instance detects potential malicious traffic and logs alerts for such activity. These logs are then forwarded to Splunk, where further action such as detailed analysis can be taken on the logged activity. Both open source and commercial network tools were able to strengthen our cloud network's security by giving us the ability to predict malicious behavior and alert the administrator of any and all incoming attacks. Thus, our NIDS simulator application provides a working example of a clear and comprehensive solution to common major network security concerns affecting cloud computing.

## REFERENCES

[1] Khalil, I., Khreishah, A. and Azeem, M., 2014. Cloud Computing Security: A Survey. Computers, 3(1), pp.1-35.

[2] LearnITGuide Tutorials. "AWS VPC | Create New VPC with Subnets, Route Tables, Security Groups, NACL | AWS Beginners Tutorial." YouTube. Available: https://youtu.be/gUesnoDzNr4. [Accessed: 10-Oct-2021].

[3] J. Ruostemaa, "How to install Snort on debian," UpCloud, 06-Nov-2020. [Online]. Available: https://upcloud.com/community/tutorials/installing-snort-on-debian/.[ Accessed: 14-Oct-2021].

[4] Rapid7, "Understanding and configuring snort rules: Rapid7 blog," Rapid7, 26-Oct-2020. [Online]. Available: https://www.rapid7.com/blog/post/2016/12/09/understanding-and-configuring-snort-rules/. [Accessed: 20-Oct-2021].

[5] FinAck, "Splunk Universal Forwarder for Linux," YouTube, 17-Oct-2021. [Online]. Available: https://www.youtube.com/watch?v=D904ON7Nu6k.

[6] Snort, "Snort FAQ," Snort. [Online]. Available: https://www.snort.org/faq/readme-filters.

[7] Kali, "hping3," Kali Linux Tools, 26-11-2021. [Online]. Available: https://www.kali.org/tools/hping3/. [Accessed: 05-Dec-2021].

[8] "HPING3 - Syn flooding, ICMP flooding &amp; land attacks," Cybarrior, 31-Jan-2019. [Online]. Available: https://cybarrior.com/blog/2019/01/31/hping3-syn-flooding-icmp-flooding-land-attacks/. [Accessed: 05-Dec-2021].

[9] Kali, "hydra," Kali Linux Tools, 26-11-2021. [Online]. Available: https://www.kali.org/tools/hydra/. [Accessed: 05-Dec-2021].]

[10] nmap.org, "Nmap Network Scanning," nmap. [Online]. Available: https://nmap.org/book/nmap-defenses-detection.html. [Accessed: 05-Dec-2021