DOI: https://doi.org/10.48009/3_iis_2022_110

Botnet intrusion detection: A modern architecture to defend a virtual private cloud

Robert, Brison, *Georgia Southern University , rb07498@georgiasouthern.edu*
Hayden Wimmer, *Georgia Southern, University, hwimmer@georgiasouthern.edu*
Carl M. Rebman, Jr., *University of San Diego, carlr@sandiego.edu*

Abstract

Advances in artificial intelligence (AI), technology integration, and cloud computing, has resulted in an increase of cybersecurity attacks by botnets over the last few years. Attackers use botnets to overwhelm and compromise networks with a goal of disrupting services or operations, stealing credentials, gaining unauthorized access to critical systems, or to obtain information for theft or ransom. The rise in this AI technology has made the job of protecting networks more challenging for network security analysis and professionals. The migration of companies and organizations into the chaotic cloud environment has really given new power to the botnets that is visualized best by scenes in any of matrix movies. One of the best methods of protection of any network or resource is early detection, which can prevent a network from being compromised or minimizing damage to the network. Two modern tools used in network security are Intrusion Detection Systems (IDS), and Security Incident and Event Management (SEIM) systems. This study proposes and tests a modern architecture to detect Botnet traffic through the implementation of modern security devices to defend against a configured local Botnet in a virtual cloud environment. Our model was successful in detecting and preventing botnet attacks. The model also allowed for the attack data to be stored and classified for report generation.

Keywords: SIEM, IDS, Splunk, Botnet, Snort, Cybersecurity, Cloud

Introduction

In recent years, there has been a considerable rise in cyberattacks on networks. Botnets are increasingly becoming one of the most widely used methods for leveraging attacks. According to a 2022 study by Imperva Research Labs 57% of all attacks on e-commerce websites were carried out by botnets. A Botnet is a network of computers or bots connected by the Internet. A bot or zombie computer is a compromised device under the control of a central attacker. The central control center used to remotely control the bots is known as a Command-and-Control Center (CnC Center). Botnets are used for cyberattacks, such as Distributed Denial of Service (DDoS) attacks, phishing, and spamming (Abraham et al., 2018; Acarali, Rajarajan, Komninos, & Herwono, 2016; Al-Jarrah et al., 2015; Aleksieva, Valchanov, & Aleksieva, 2019; Asha, Harsha, & Soniya, 2016; Bansal & Mahapatra, 2017; Chen, Chen, & Tzeng, 2018; Haddadi, Phan, & Zincir-Heywood, 2016; Kaur & Singh, 2016; Shetu, Saifuzzaman, Moon, & Nur, 2019; Syamsuddin & Barukab, 2022; Ujjan, Pervez, & Dahal, 2019; Vacas, Medeiros, & Neves, 2018) .

Botnets and the cyberattacks deployed are evolving to bypass traditional or legacy security systems. In past iterations, cyberattacks contain a malicious digital signature. Traditional systems would be trained to recognize these digital signatures, alert security analysts, log the threat, and block it from entering the system. This model was sufficient to protect business networks from external attacks. Cyberattacks have become more sophisticated with the usage of unknown signatures, allowing them to circumvent traditional security models (Ujjan et al., 2019).

Traditional firewalls and Intrusion Detection Systems (IDS) alone are becoming more obsolete in meeting modern security needs (Ujjan et al.). Security architectures created by legacy devices need to be updated to better defend against the threat of Botnets. One modern security device used to manage a security architecture is known as Security Incident and Event Management (SIEM) systems. These devices are used for log management and to provide security analysts with capabilities to investigate events in real-time. SIEMs provide support for security devices by providing the devices a centralized command center.

Organizations have been migrating business models to a decentralized, remote structure otherwise known as a virtual cloud environment. Outsourcing resources into the cloud has benefits as it allows for employees to access resources remotely and it allows for organizations to grow globally and operate continuously. A common model implemented for virtual cloud environments is the implementation of VMware's vSphere joined with the Elastic Sky X (ESXi) server. The ESXi server operates as a bare metal solution to provide the semblance of hardware for virtual resources to operate on. The ESXi server is managed by the vSphere client platform. This model is a cost-efficient method to implement a virtual cloud environment (Nomnga, Scott, & Nyambi, 2014).

Virtual cloud environments have been a target for Botnets. In this paper, we have created an architecture designed to detect Botnet attacks through the connection of an IDS with a custom, updated ruleset to a SIEM. This architecture allows for alerts, logging, and analysis of Botnet attacks in real-time for network security analysts to mitigate against the attack. This paper is organized as follows. Section II reviews research concerning SIEM and IDS capabilities and different forms of Botnet detection. Section III covers background information of Botnets, SIEMs and IDS/IPS devices. Section IV describes the methodologies used in this experiment. Section V displays the results of this experiment. Section VI discussed the results and implication of the experiment. Section VII end with a summary and conclusion.

Literature Review

This section discussion the terminology and research studies concerning two main tools in network security: specifically, Security Incident and Event Management (SIEM), Intrusion Detection Systems (IDS), and form of botnet detection.

SIEM Defense

Given SIEMs roles in network defense architecture, applying this technology to other applications is the progression of its abilities. Hwoij, Khamaiseh, and Ababneh (2021) present a potential model to defend the expansion of IoT devices. IoT is projected to create a smart network through information gathering sensors to create a smart home. The extension of the smart home is expected to result in the creation of smart cities. Large networks with multiple entry points for attackers will require an expansive defense architecture to become viable.

Hwoij et al. (2021) sought to create relationships between the logs of IoT devices with SIEMs to display anomalies and cyber threats, though SIEMs performance of data correlation and data analysis by log aggregation. (Hwoij et al., 2021) used the Splunk SIEM to create a three-step architecture that protects smart cities. The proposed architecture is made up of three steps. Step one is the smart environment made up of the different smart devices. Step two is the Splunk SIEM. Step three is the security operation center. The architecture relies on forwarders, which receive data from the nodes or smart devices, which in turn is indexed by Splunk. The results wase a comprehensive model with the ability to provide real-time monitoring, investigation abilities, and threat hunting capabilities.

Machine Learning for IDS

Machine Learning (ML) has the potential to advance cybersecurity through novel threat detection and signature-based learning. Abubakar and Pranggono (2017) and Shah and Issac (2018) provide different applications through different approaches of ML to IDS technology. There are numerous IDSs available for network security. Two of the most widely used open-source IDSs are Snort and Suricata. Shah and Issac (2018) perform a series of tests to determine the more suited IDS for an integrated ML approach. Both Snort and Suricata detect malicious traffic based on rulesets but are unable to detect unknown attacks. Integrated ML technologies can assist IDSs detect unknown attacks not found in their rulesets. The researchers seek to measure the accuracy of both IDSs and to evaluate the impact of false positives after the integration of different ML techniques.

Shah and Issac (2018) used seven different types of malicious traffic with rules that could be applied to both Snort and Suricata. The experiments compared the performance of both IDSs by their usage of CPU, memory usage, and rate of packet dropping. Experiment traffic was created by three different open-source generators, while the malicious traffic was created by Metasploit. The default rulesets were used by both, and a sandbox was created by Oracle Virtual Box. Three scenarios were experimented. The first scenario measured real-time performance of the IDSs while processing 10 Gbps network speed from one of the legitimate network traffic generators. Tools were used to measure the CPU usage, memory, and packet drop rate. Scenario two measured the accuracy of the IDSs to classify legitimate and illegitimate traffic. Test two measured the false positives, false negatives, and true positives rates of both legitimate and illegitimate traffic. Metasploit injected UDP, TCP, and ICMP packets into the IDSs. The third scenario tested Snort integrated with additional plug-ins. The first used Individual Algorithms and the second used Hybrid and Optimized Algorithms.

Shah and Issac (2018) found Snort was less computational but had lower CPU usage and fewer false positives. Suricata had more computational power but also required more CPU usage with more false positives. Suricata was able to process 10 Gbps with lower packet drop rates than Snort. Out of the seven types of malicious traffic tested, Suricata detected four while Snort detected six. The three plug-ins integrated into Snort improved its functionality, but the integration of the Support Vector Machine with the firefly algorithm gave the lowest false positive and negative rates. This lead (Shah & Issac, 2018) to determine Snort is the most suited to handle ML technology. It is expected to yield the best results with the lowest strain on network resources.

Software Defined Networks (SDN) are potential advances for designing and building network architectures. Abubakar and Pranggono (2017) use an integrated ML-IDS into SDNs to create a potential network security architecture. SDNs allow for network control to be programmable. SDNs are comprised of a three-tiered architecture to manage a network. Tier one is the Application layer, tier two is the SDN Controller, and tier three Infrastructure Layer. Traditional network architecture is becoming more complex and rigid for modern devices. In standalone environments, SDNs have security holes, but the researchers propose integrating an IDS to enhance security.

Abubakar and Pranggono (2017) created a virtual sandbox to test the SDN/Snort architecture using different attacks against both servers and normal users. The virtual sandbox was set up using Ubuntu OS and controlled by an OpenDayLight controller. The Snort IDS is installed on a second Ubuntu VM. Two other VMs were created in the sandbox. One is for a network security toolkit to leverage attacks, and the second is the server the researchers are going to attack. Wireshark is used to monitor network traffic. A second method measured by the researchers employed a NN model for anomaly detection through pattern recognition. A backpropagation algorithm is used to train the NN to classify attacks. The NSL-KDD dataset was used to train the model. The model showed a high detection accuracy for the training and testing sets. The pattern recognition of the NN had a better accuracy for detection. The results show integrating Snort and an NN to defend SDNs.

DDoS Detection Using Splunk

Distributed Denial of Service (DDoS) are one major type of cyberattacks that continues to target networks and devices. DDoS attacks can be damaging to a network or to devices because the rapid inflow of traffic can take devices or networks offline, hindering business operations. Hristov, Nenova, Iliev, and Avresky (2021) and Su, Wang, Chen, and Liu (2016) took different approaches to test Splunk's ability to detect DDoS attacks on a network.

In modern security models, the security at the perimeter is no longer sufficient for network security. Hristov et al. (2021) studied using a Splunk SIEM to protect against malware. SIEMs collect logs from devices and the logs are forwarded to Splunk. Raw data is indexed in Splunk. It parsed and sent into indexes, where it can be searched for later.

Hristov et al. (2021) set up two PCs for their experiment. The first PC had Splunk installed on it. The Splunk PC was configured to have logs from a firewall sent to it. (Hristov et al., 2021) created four rules to trigger alerts in Splunk. The first rule monitors successful logins into Splunk. The second rule flags communication with the Command-and-Control server. The third rule monitors when the security log is cleared. The fourth rule logs every failed login attempt. To test their model, Hristov et al. (2021) used the Mirai Botnet malware to perform DDoS attacks. The configuration was able to detect connection to the CnC server in Splunk. The configuration was able to detect the IP address of the malicious activity.

Su et al. (2016) took a different approach to test Splunk's ability to identify, log, and display DDoS attacks in networks. Su et al. (2016) used the Kali Linux tools Hping3 and Scapy to mimic a DDoS attack. The two tools combined to create a flood, reflect, and amplify the attack. Wireshark was used to collect network traffic for the attack and the network traffic was then be passed to Splunk for analysis. Splunk was used to display the network traffic for DDoS signatures to be identified through different visualizations.

Splunk was also integrated with Google Maps to present locations of traffic. Su et al. (2016) found that Splunk was able to display the traffic showing DDoS attacks and source location through Google Maps. Chart analysis was used to discover hidden risks and detect potential attacks. Splunk was able to determine if an attack had occurred. Splunk was able to store data from different attacks. This use of Splunk demonstrates it is an effective IDS for network

Botnet Detection

The renewed usage of Botnets for cyberattacks prompted several methods to be proposed for Botnet detection. A common, passive method used is the Honeypot approach. Honeypots are a faux computer system created to attract cyberthreats. Honeypots contain different applications, data, etc. to mimic legitimate systems, but they have built-in vulnerabilities to draw attention of attackers. Once the attacker connects with the Honeypot, the cybersecurity analyst can learn about the threats of the attacker. While Honeypots offer some benefits to learn about different potential attacks, they do have some risk associated with them. If an attacker realizes there is a Honeypot, it can be leveraged as a way to tunnel into a network or it can be fingerprinted in order to exploit the actual system (Shetu et al., 2019).

Other approaches have been developed to detect Botnets. Several of these approaches have centered around implementation of machine language (ML) techniques to identify Botnet attacks. Botnets have become supplemented by the integration of AI principles and contain unknown signatures to bypass traditional firewall and IDS/IPS devices. ML in theory should be capable of handling larger amounts of volume from attacks, as well as having the capability to learn unknown attack signatures.

Bansal and Mahapatra (2017) compared three methods for using different ML techniques to detect Botnet activity using two updated datasets. The first method is the Clustering method. The second method used in the study was a Neural Network (NN) Method. In the NN method, the data is bucketed, processed, and ran through a NN created based on Long Short-Term Memory (LSTM) cells. The third method in the study is a Recurrent Neural Network (RNN). The RNN differs from the NN in the previous method in that it removed the manual step from data inputs. Bansal and Mahapatra (2017) found the Clustering method had a high accuracy with a F1 score of 0.8545. The NN method was displayed to have a good accuracy overall with a F1 score of 0.8897. The RNN had a lower accuracy with a F1 score of 0.8044. From the results, the NN approach showed the most promising results for the three tested ML techniques.

Chen et al. (2018) proposed a different ML technique for detecting Botnet traffic. The model used connects a Convolutional Neural Network (CNN) with an Artificial Neural Network (ANN) to detect P2P Botnets. The model follows a three-stage process. The first is the processing stage where data is defined, compressed, and computed. The data is then pushed through the training stage, where it is passed through a CNN. Also, during this stage, an ANN is fully connected. The final stage is the Confidence testing phase. During this phase, the results from the ANN are classified based on confidence. Three experiments were ran measuring the Parameter Selection, the Effectiveness of the Training Stage, and the Effectiveness of the Confidence stage. The results from the experiment showed promising results for detecting P2P Botnets with high accuracy and low FPRs.

Botnets in the Cloud

There has been a focus of Botnets to target cloud environments. This shift has mirrored the migration of organizations to cloud environments. Koroniotis, Moustafa, Sitnikova, and Turnbull (2019) seek to address this issue by creating a dataset to train network security in the cloud environment. The proposed dataset, or the “Bot-IoT” encompasses simulated IoT network traffic along with different attacks. The architecture uses VMs in a virtual cloud environment hosted by an ESXi server that is managed by a vSphere client. In the architecture, an external packet-filtering firewall monitors all incoming traffic. Traffic is then passed through a VMware cluster into a switch and then through a firewall to enter the virtual cloud network.

In the model by Koroniotis et al. (2019), a dataset is compiled of the various attacks commonly used by Botnets against networks. The data from the simulated traffic is collected in pcap files and imported into a MySQL database for processing. After processing, Koroniotis et al. (2019) were able to create a viable dataset to train network security architectures in cloud environments for common Botnet attacks.

Background Information of Botnet, SIEM, IDS

Botnets

Attackers use different methods to penetrate networks. The tools available are ever evolving, with the implementation of technology such as AI, and signature masking. Integrating these technologies into Botnets has caused a resurgence of Botnets as cyber threats. In creation, Botnets require a multistep process. Botnets follow a general five phase lifecycle:

- ***Infection Phase-*** The first phase is the infection phase. During this phase, the attacker recruits other devices to become bots. Weaknesses in potential devices are exploited by attackers to infect the devices through malware.
- ***Injection Phase*** The second phase is the injection phase. During this phase the infected devices execute programs or scripts from a network database. Once executed, these scripts or programs turn the device into a bot to be used by the Botnet.
- ***Connection Phase-*** The third phase is the connection phase. During this phase, the newly recruited bot locates and connects to the CnC server. Once connected, the bot can then receive and execute commands from the botmaster through the CnC channel.
- ***Malicious Phase*** The fourth phase is the malicious phase. During this phase, the bot carries out the attacks sent from the botmaster remotely. The bot performs malicious attacks, such as DDoS and spamming.
- ***Maintenance Phase-*** The fifth and final step in the Botnet lifecycle is the maintenance phase. During this phase, the botmaster seeks to keep recruited bots under its control. The botmaster needs to continually update its network of bots with new capabilities. (Asha et al., 2016; Kaur & Singh, 2016; Shetu et al., 2019).

The communication channel between the botmaster and its bots follows a particular architecture. The traditional communication channel used Internet Relay Chat (IRC). Under this architecture, the Botnet communicates over IRC using internet text messages. While this architecture is flexible, its traffic is easily traced. Another architecture used in communication channels is HTTP. This allows for Botnet traffic to be hidden along with normal traffic. This allows for an easier bypass of network firewalls. Another architecture used in communication channels is a Peer-to-peer model. In this model, bots relate to each other in order to hide the CnC server. It is a more complicated architecture, but makes tracing traffic more difficult (Kaur & Singh, 2016).

SIEM

One of the most common approaches to modern network security is the implementation of Security Information and Event Management (SIEM) systems (Hristov et al.). An SIEM is a centralized set of security software tools used to collect, store, log, and report data in a network. SIEMs combine Security Information Management (SIM) and Security Event Management (SEM) principles (Cinque, Cotroneo, & Pecchia, 2018; Mokalled et al., 2019; Sekharan & Kandasamy, 2017). SIMs are tools used to collect log file data to analyze and report security threats and events. SEMs are tools used for real-time system monitoring to notify the network administrators of security events and establish correlations between events. SIEMs gathers network data and makes it available to network administrators for analysis. A SIEMs architecture can be made up of firewalls, IDS/IPS, anti-virus software, etc. (Hristov et al., 2021; Sekharan & Kandasamy, 2017). SIEMs is a central location for every device's network logs to be sent to, stored, and analyzed. SIEMs are able to handle and process large datasets generated from network traffic logs with the ability to offer statistics in different visualizations, making them desirable tools for businesses (Sönmez & Günel, 2018). The integration of SIEMs for organizations has its own challenges (Mokalled et al., 2019). SIEMs require large amounts of resources and configuration to ensure logs are correctly handled, and the rulesets the SIEM operates from are properly configured to prevent false positives or negatives for network traffic (Cinque et al., 2018).

IDS/IPS

Network defense architectures are comprised of different devices. An Intrusion Detection System (IDS) or Intrusion Prevention System (IPS) are one of the devices found in architectures. An IDS is a device used to monitor networks or systems for abnormal or malicious activity that violates the rules or policies of an organization (Kenaza & Aiash, 2016). Any violation detected by the IDS is sent to the network admin or to the SIEM to be determined if the event is malicious or not. IDS differs from firewalls in that firewalls prevent outside traffic based on rulesets. IDS take the next step to alert of suspicious or malicious activity. IDS also can monitor what is going on within a network to monitor potential inhouse attacks or violations. IDS detect events or intrusions by analyzing specific patterns of traffic or sequences. The common placements of IDS are behind the network firewall to give the system high traffic visibility as it enters the network (Shah & Issac).

IDSs have two different modes of operation: passive and active. A passive IDS alerts and logs incoming threats to a network. An active IDS does these along with giving the IDS the ability to document the threat (Kumawat, Sharma, & Kumawat, 2016). An IDS has different implementation options with different defense capabilities. An IDS can be configured to detect malicious traffic from a signature-based, anomaly-based, network-based, or host-based approach. A signature-based implementation configures the IDS to detect threats based on a matched signature in the known base. If traffic matches a known signature, the IDS is alerted. An anomaly-based implementation configures the IDS with a set baseline. Any deviation from this baseline triggers the IDS. An anomaly-based IDS is regularly evolving from learned data (Abraham et al., 2018). A network-based implementation monitors all traffic coming through the network and relies on filters to detect malicious traffic. A host-based IDS implementation uses a host to audit incoming traffic. The audit uses different methods to detect malicious traffic (Kumawat et al., 2016).

The advancement of IDS technology has evolved into IPS. An IPS is a device used to identify malicious activity, log the activity, report the activity, and stop the activity. IPS attempts to block any intrusions found in the network by taking actions such as dropping flagged packets, sending alarms, and blocking malicious IP addresses. An IPS differs from the older IDS in that it is a control system. An IDS is limited in its ability to monitor packet content, while an IPS typically can block traffic based on packet contents. An IPS is a more active security device compared to the IDS. Either system can be integrated into a network and into a SIEM for network security.

Methodology

Architecture

The research was performed in a local research lab with a dedicated network. On this local network, a virtual sandbox was created using VMware. Three virtual machines (VM) were created in this sandbox. The first VM was created using the Ubuntu 20.04 OS. On this VM, Snort 2.9.18.1 was installed with custom rules to detect ICMP and TCP requests on the local network. Snort was used in signature-based detection mode. This was done as opposed to anomaly-based detection because anomaly-based detection settings can have higher false-positive rates than signature-based detection settings (Syamsuddin & Barukab, 2022).

The second VM was also created using the Ubuntu 20.04 OS. On this VM, Splunk Enterprise version 8.2.6 was installed. The 64-bit .deb version for Linux was chosen for this research. The instance of Splunk was configured to monitor the local network, and a Snort plugin was installed on it. Figure 1 illustrates how a forwarder was installed on the Snort VM to pass traffic logs to the Splunk VM.

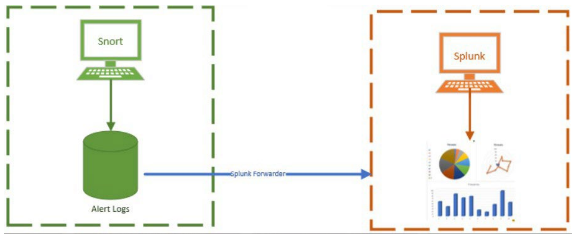


Figure 1 Forwarding traffic logs to VM

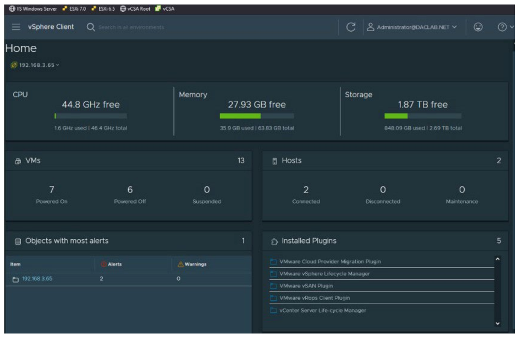
In the sandbox, the Snort architecture was tested by the Kali Linux VM. The Kali Linux VM pinged the Snort VM to ensure Snort was able to intercept traffic. The Kali Linux tool hping3 was used to simulate a DDoS attack. Snort was able to alert the incoming attack. Once the Snort VM was shown to work in the limited environment, the sandbox was migrated to a local ESXi server to host the environment. Our architecture differs from (Koroniotis et al., 2019), in that we do not have an external packet-filtering firewall. The model present here is not designed to create a dataset from training, rather to display an architecture capable of detecting and investigating Botnet attacks.

Botnet

An open-source Botnet called Bring Your Own Botnet (BYOB) was used in the research. A VM was created using Ubuntu 20.04 OS to host the Botnet. The BYOB follows a client-server architecture and communicates over an HTTP communication channel. The BYOB was adapted to run in the local environment set up in the local lab. The BYOB was configured for the bots to use their own IP addresses to prevent mixing of local IP addresses in the lab and local IP addresses from other nearby networks. The BYOB was configured to launch port scanning and DDoS attacks. During the testing phase of the BYOB, traffic was captured in Wireshark. The traffic from the Botnet was analyzed to make sure it was recruiting bots to attack the network. These packets captured in Wireshark were inspired in the creation of rules in Snort. Bots were recruited to join BYOB by installing and executing a Python file. Once the Python file is executed, a payload is downloaded and executed on the target machine as well. After execution, the target machine becomes a bot in the BYOB. At this point, the bots are under the control of the CnC. Once the BYOB was tested and operational, it was migrated onto the ESXi server.

SXi Server

In the local research lab, an ESXi server was set up. ESXi is a VMware product we installed on a local server. The ESXi server was used in conjunction with vSphere to create a virtual private cloud for the experiment. The ESXi is managed by vSphere. The vSphere Client allows for VMs to be added, managed, and created within the ESXi server. Figure 2 illustrates the sandbox for the security architecture and the BYOB were uploaded from VMware.



Home

192.168.3.65

CPU	Memory	Storage
44.8 GHz free	27.93 GB free	1.87 TB free
1.6 GHz used | 46.4 GHz total	35.9 GB used | 63.83 GB total	848.09 GB used | 2.69 TB total

VMs 13

Powered On	Powered Off	Suspended
7	6	0

Hosts 2

Connected	Disconnected	Maintenance
2	0	0

Objects with most alerts 1

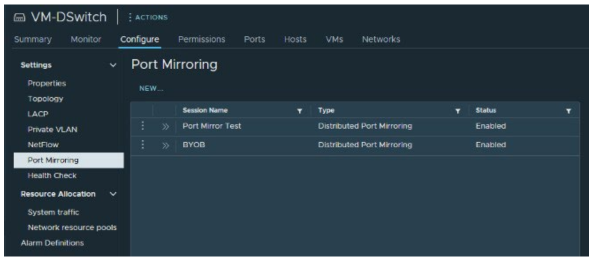
Item	Alerts	Warnings
192.168.3.65	2	0

Installed Plugins 5

- VMware Cloud Provider Migration Plugin
- VMware vSphere Lifecycle Manager
- VMware vSAN Plugin
- VMware vRops Client Plugin
- vCenter Server Life-cycle Manager

Figure 2 vShpere console hosting sandbox

In the home dashboard displayed above, the VMs are controlled remotely. Within the server, port mirroring was performed. This was done to forward network traffic, including traffic coming into the network from the BYOB, into Snort. This mimics the typical network architecture of all traffic going through an IDS as it enters the network and is shown in Figure 3.



Port Mirroring

Session Name	Type	Status
Port Mirror Test	Distributed Port Mirroring	Enabled
BYOB	Distributed Port Mirroring	Enabled

Figure 3 Port mirroring

In the VM Switch settings, the traffic from the switch is configured to be mirrored to the IDS. Once this configuration was complete, the architecture was finalized and illustrated in Figure 4. The BYOB traffic is routed through Snort. Snort creates alerts as the traffic targets the network, and this data is sent to Splunk. Splunk is then interacted with by the Network Security Analyst to generate reports to display the BYOB traffic.

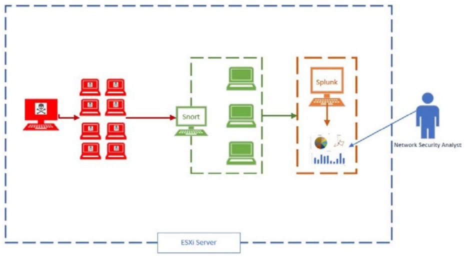
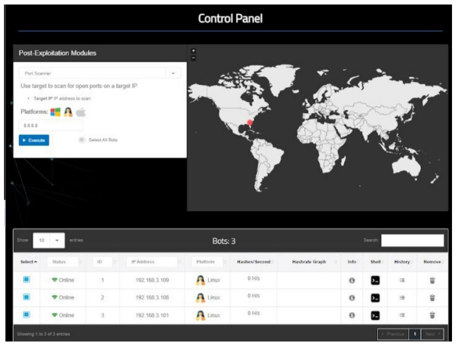


Figure 4 Migrated architecture VmSwitch to IDS

Results

The BYOB was able to attack the network. The BYOB was able to run port scanning attacks across the network using recruited bots. The BYOB was also able to DDoS the network. Figure 5 illustrates how bots were able to send traffic to the network attempting to create TCP connections.



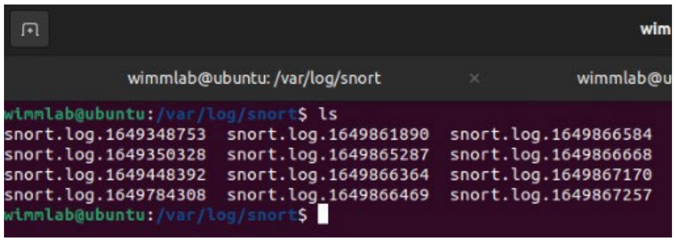
Bots: 3

Select	Status	ID	IP Address	Platform	Hashes/Second	Hashrate Graph	Info	Shell	History	Remove
	Online	1	192.168.3.109	Linux	0 H/s					
	Online	2	192.168.3.108	Linux	0 H/s					
	Online	3	192.168.3.101	Linux	0 H/s					

Figure 5 Bots sending traffic and create TCP connections

Snort

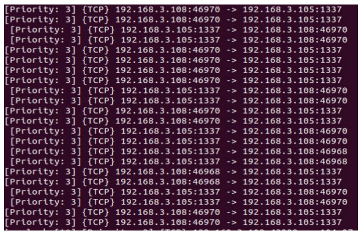
The rules in Snort were able to detect TCP and ICMP traffic coming into the network. The alerts triggered by the incoming traffic were printed to the console displaying timestamps, a descriptive message, the threat level, and the IP addresses of the connections. These alerts were stored in packet capture (pcap) files within Snort. Figure 6 displays the storage of these pcap files in Snort:



```
wimmlab@ubuntu:/var/log/snort$ ls
snort.log.1649348753  snort.log.1649861890  snort.log.1649866584
snort.log.1649350328  snort.log.1649865287  snort.log.1649866668
snort.log.1649448392  snort.log.1649866364  snort.log.1649867170
snort.log.1649784308  snort.log.1649866469  snort.log.1649867257
wimmlab@ubuntu:/var/log/snort$
```

Figure 6 Storage of pcap files in Snort

Within these pcap files, Snort stores logs where it detects both the port scanning and DDoS traffic from BYOB through the custom rules. Figure 7 displays the printed alerts from Botnet attacks within Snort: The Date and timestamp are the first part of the alert. Then the number of the rule violated followed by the alert message is next in the alert. The threat priority, type of connection, and the IP addresses finish the alert in Snort. The files were forwarded into Splunk through the Splunk Forwarder for further analysis of the Botnet traffic.



```
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46968
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46968
[Priority: 3] {TCP} 192.168.3.108:46968 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46968 -> 192.168.3.105:1337
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
 [Priority: 3] {TCP} 192.168.3.105:1337 -> 192.168.3.108:46970
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
[Priority: 3] {TCP} 192.168.3.108:46970 -> 192.168.3.105:1337
```

Figure 7Printed Alerts in Snort

Splunk

Splunk was logged onto by connecting to the local host on port 8000 and then a basic search was run to verify data was coming from Snort into Splunk. Once it was verified data was coming into Splunk, the Snort for Splunk application was downloaded and installed. In the Snort for Splunk app, data from the Splunk Forwarder was pulled in to generate a dashboard called Snort event summary and is shown in Figure 8.

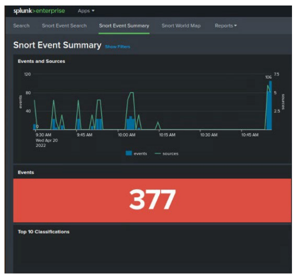
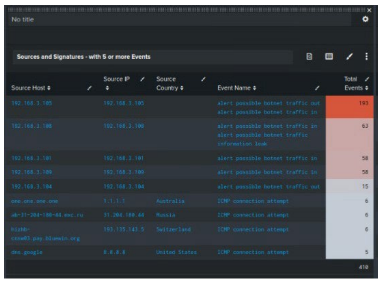


Figure 8 Snort Event Summary

Bots from different countries flooded the network with TCP connections as displayed in the Event Chart. The country of origins of the bots is displayed in the Pie Chart, broken down by the number of attacks by bot. Figure 9 illustrates the dashboard and how the sources and volume of events are found by drilling down on the events. The source IP addresses, country of origin, and the event names are displayed in these logs. The frequency of each event is displayed in the farthest right column. Splunk was able to receive the BYOB traffic logged by Snort.


No title

Sources and Signatures - with 5 or more Events

Source Host	Source IP	Source Country	Event Name	Total Events
192.168.3.105	192.168.3.105		alert possible botnet traffic out alert possible botnet traffic in	193
192.168.3.108	192.168.3.108		alert possible botnet traffic in alert possible botnet traffic information leak	63
192.168.3.101	192.168.3.101		alert possible botnet traffic in	58
192.168.3.109	192.168.3.109		alert possible botnet traffic in	58
192.168.3.104	192.168.3.104		alert possible botnet traffic out	15
one.one.one.one	1.1.1.1	Australia	ICMP connection attempt	6
ab-31-204-180-44.mxc.ru	31.204.180.44	Russia	ICMP connection attempt	6
hizhb-cnsw03.pay.bluewin.org	193.135.143.5	Switzerland	ICMP connection attempt	6
dns.google	8.8.8.8	United States	ICMP connection attempt	5
				410

Figure 9 Sources and Volumes of Attacks Dashboard

Discussion

Botnets are becoming an increasing threat to networks. Detecting them early gives network security analysts an advantage to block them from the network before the network becomes compromised. With the integration of networks into virtual cloud environments and the targeting of these environments by Botnets, an architecture is needed for protection. The proposed architecture gives network security analysts this ability. All the incoming traffic is sent through Snort by port mirroring, so Snort can process all network traffic. In doing so, our configuration of Snort provides alerts when the Botnet attempts to attack the network. The alerts can be viewed remotely by accessing the ESXi server. The alerts are also logged into pcap files and forwarded to Splunk. Once the alerts in Snort notify of the incoming Botnet attack, Splunk can also be connected to remotely to investigate the attack. This gives the network security analyst the ability to see where the attack is coming from, the malicious IP addresses, and the ability to drill down in reports to learn about the attack. From here, the network security analyst can counter the attack in a more effective way.

Conclusion

We proposed a model designed to detect Botnet traffic coming into a virtual cloud network. Our model created an IDS configured to detect incoming traffic used by Botnets. Also in the IDS, a forwarder was installed to send data from the IDS along in the model. It included a SIEM to interact with the logged data from the IDS. Through this integrated model, malicious traffic can be investigated to learn about the attack. Our model allows for the attack data to be stored and for reports to be generated from the attack data. These reports can be used to find out important information about the attack, such as the volume of traffic, source IP addresses, connection protocol, etc.

In further research, more devices can be integrated into our architecture. Firewalls, for instance, can be integrated to filter out malicious traffic from a network, and these logs can be sent to the SIEM as well to give a fuller picture of network cybersecurity. Another potential addition to the architecture is the implementation of a Demilitarized Zone to further segment the network. The use of a DMZ can allow for security logs to remain away from the sensitive information in a network while also sending malicious attack logs to the SIEM for investigation. The expansion of our model could allow for a wider toolkit for network security analysts to protect networks. We conclude that our model presents a viable option for countering Botnet attacks, and it gives network security analysts the tools to block these attacks in virtual cloud networks.

References

Abraham, B., Mandya, A., Bapat, R., Alali, F., Brown, D. E., & Veeraraghavan, M. (2018). *A comparison of machine learning approaches to detect botnet traffic.* Paper presented at the 2018 International Joint Conference on Neural Networks (IJCNN).

Abubakar, A., & Pranggono, B. (2017). *Machine learning based intrusion detection system for software defined networks.* Paper presented at the 2017 seventh international conference on emerging security technologies (EST).

Acarali, D., Rajarajan, M., Komninos, N., & Herwono, I. (2016). Survey of approaches and features for the identification of HTTP-based botnet traffic. *Journal of network and computer applications, 76*, 1-15.

Al-Jarrah, O. Y., Alhussein, O., Yoo, P. D., Muhaidat, S., Taha, K., & Kim, K. (2015). Data randomization and cluster-based partitioning for botnet intrusion detection. *IEEE transactions on cybernetics, 46*(8), 1796-1806.

Aleksieva, Y., Valchanov, H., & Aleksieva, V. (2019). *An approach for host based botnet detection system.* Paper presented at the 2019 16th Conference on Electrical Machines, Drives and Power Systems (ELMA).

Asha, S., Harsha, T., & Soniya, B. (2016). *Analysis on botnet detection techniques.* Paper presented at the 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS).

Bansal, A., & Mahapatra, S. (2017). *A comparative analysis of machine learning techniques for botnet detection.* Paper presented at the Proceedings of the 10th International Conference on Security of Information and Networks.

Chen, S.-C., Chen, Y.-R., & Tzeng, W.-G. (2018). *Effective botnet detection through neural networks on convolutional features.* Paper presented at the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).

Cinque, M., Cotroneo, D., & Pecchia, A. (2018). *Challenges and directions in security information and event management (SIEM).* Paper presented at the 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW).

Haddadi, F., Phan, D.-T., & Zincir-Heywood, A. N. (2016). *How to choose from different botnet detection systems?* Paper presented at the NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium.

Hristov, M., Nenova, M., Iliev, G., & Avresky, D. (2021). *Integration of Splunk Enterprise SIEM for DDoS Attack Detection in IoT.* Paper presented at the 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA).

Hwoij, A., Khamaiseh, A. h., & Ababneh, M. (2021). *SIEM architecture for the internet of things and smart city.* Paper presented at the International Conference on Data Science, E-learning and Information Systems 2021.

Kaur, N., & Singh, M. (2016). *Botnet and botnet detection techniques in cyber realm.* Paper presented at the 2016 International Conference on Inventive Computation Technologies (ICICT).

Kenaza, T., & Aiash, M. (2016). Toward an efficient ontology-based event correlation in SIEM. *Procedia Computer Science, 83*, 139-146.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems, 100*, 779-796.

Kumawat, S., Sharma, A. K., & Kumawat, A. (2016). *Intrusion detection and prevention system using K-learning classification in cloud.* Paper presented at the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom).

Mokalled, H., Catelli, R., Casola, V., Debertol, D., Meda, E., & Zunino, R. (2019). *The applicability of a siem solution: Requirements and evaluation.* Paper presented at the 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE).

Nomnga, P., Scott, M., & Nyambi, P. (2014). A technical cost effective network-domain hosting through virtualization: a VMware ESXi and vSphere client approach. *International Journal of Computer Applications, 975*, 8887.

Sekharan, S. S., & Kandasamy, K. (2017). *Profiling SIEM tools and correlation engines for security analytics.* Paper presented at the 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET).

Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems, 80*, 157-170.

Shetu, S. F., Saifuzzaman, M., Moon, N. N., & Nur, F. N. (2019). *A survey of Botnet in cyber security.* Paper presented at the 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT).

Sönmez, F. Ö., & Günel, B. (2018). *Evaluation of security information and event management systems for custom security visualization generation.* Paper presented at the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT).

Su, T.-J., Wang, S.-M., Chen, Y.-F., & Liu, C.-L. (2016). *Attack detection of distributed denial of service based on Splunk.* Paper presented at the 2016 International Conference on Advanced Materials for Science and Engineering (ICAMSE).

Syamsuddin, I., & Barukab, O. M. (2022). SUKRY: Suricata IDS with Enhanced kNN Algorithm on Raspberry Pi for Classifying IoT Botnet Attacks. *Electronics, 11*(5), 737.

Ujjan, R. M. A., Pervez, Z., & Dahal, K. (2019). *Snort based collaborative intrusion detection system using blockchain in SDN.* Paper presented at the 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA).

Vacas, I., Medeiros, I., & Neves, N. (2018). *Detecting network threats using OSINT knowledge-based IDS.* Paper presented at the 2018 14th European Dependable Computing Conference (EDCC).