MITIGATION OF CROSSFIRE ATTACKS IN SOFTWARE DEFINED NETWORKS USING RANDOM FOREST MACHINE LEARNING ALGORITHM

by

Gabon Williams

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Cybersecurity

Charlotte

2020

Approved by:

_____

Dr.Weichao Wang

_____

Dr.Thomas Moyer

_____

Dr.Heather Lipford

ABSTRACT

GABON WILLIAMS. Mitigation of Crossfire Attacks in Software Defined Networks Using Random Forest Machine Learning Algorithm. (Under the direction of DR.WEICHAO WANG)

Software Defined Networks have emerged and developed to become a prevalent industry used infrastructure. The mitigation around DDoS (Distributed Denial of Service) attacks in SDN has been a big topic since the network type has come into the scope of day to day operations. The crossfire attack is a link flooding Distributed Denial of Service attack that increases the amount of benign traffic from a massively distributed botnet to congest a network link, also known as the target link. This spike in traffic is used to deplete the network resources allocated for the target link. This attack normally does not contain any malicious payloads, which makes detecting and mitigating more difficult. This research was inspired by the low likelihood of the ability to detect and defend against a crossfire attack in software defined networks and is focused on detection and mitigation of these attacks [1]. The environment created in this experiment uses SDN Switches on a specific network topology in addition to the ryu controller. The random forest machine learning model was also utilized to dynamically analyze traffic and classify when an attack was beginning to occur. When the classifier alerts the controller that the threshold is being reached for a particular target link the controller will find and deny flows from the source which has generated the most traffic and is not regularly generating traffic at such a high rate. Denying flows closer to the source limits the impact that a flood can have on legitimate traffic in the topology. The threshold set for the target link in this experiment is 50 MB. The model is able to classify when an interface needs to be denied at a rate of 90.625 percent. Considering this is a base work, accuracy could be improved by having additional functionality applied to the proposed design.

DEDICATION

I would like to dedicate this work to the Father I serve for without him this would not be possible. I dedicate this to all the youth struggling in their under privileged communities as I have in the past. I dedicate this work to my family and the people who helped me along this journey. Without all of their support I would not be here today and this works' creation would not have been possible.

ACKNOWLEDGEMENTS

I would like to acknowledge the role of my advisory committee and the role that my advisor Professor Wang played in allowing for the development of this thesis over time. I would also like to acknowledge everyone who had a hand in making this a possibility.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACK An acronym for Acknowledgement.

API An acronym for Application Programming Interface.

C2 An acronym for Command and Control.

CGI An acronym for Computer Generated Imagery.

DDOS An acronym for Distributed Denial of Service

IDS An acronym for Intrusion Detection System.

IP An acronym for Internet Protocol.

IPS An acronym for Intrusion Prevention System.

ITS An acronym for Intelligent Transport Systems.

LAN An acronym for Local Area Network.

LFA An acronym for Link Flooding Attack.

OF An acronym for OpenFlow

REST An acronym for Representational State Transfer.

SDN An acronym for Software Defined Networks.

SIS An acronym for Software and Information Systems.

VM An acronym for Virtual Machine.

CHAPTER 1: INTRODUCTION: THE CONCEPTS

The motivation behind this work includes the fast development of SDNs, the unique challenges to SDN security and the potential impacts of attacks on SDNs. The International Data Corporation states that the SDN market will be worth more than twelve billion dollars in 2022. The developments of SDN in industries such as protection against misconfigurations of regular networks, its utilization with self-driving cars and 5G are reasons why the popularity of SDN is growing so fast. However, with new technologies come new vulnerabilities and new attack surfaces. Attackers are coming up with ways to exploit the protocol weaknesses in SDN. Attackers are able to trigger information disclosure inside these networks that can then be used to exploit other components of the network. The potential impacts of these attacks include compromise of the network and exhaustion of resources. With attackers becoming more sophisticated with C2 Frameworks and DDoS attacks moving inside the internal network by compromising internal hosts, this research has become necessary in mitigating such attacks from occurring. The following chapter introduces relevant background in regards to the concepts discussed in this research. The topic of SDN, the link flooding attack and machine learning will be introduced.

1.1 Network Technology

Development of networking technology over the past few years has been tremendous in terms of creating substantial resources in order to supply for business needs. Traditional networks of the past found it hard to support the fast development of business applications. Network devices were responsible for the control and forwarding of traffic but the configuration could not be changed on the fly and it was highly

cost inefficient. Additionally, with the emergence of cloud computing and storage and its reliance on flexibility from the network perspective there has been a demand of making network technology much more flexible. The need for speed is a driving reason why software defined networking (SDN) has become so prevalent in today's society. SDN is utilized in data centers, intelligent transport systems and in most cloud environments in order to allow for scalable systems in which traditional networks are not able to offer a programmatic and swiftly fixable network environment. SDN is able to virtualize the physical network in order to provide a more elastic and scalable network. Cloud environments are causing a shift in network technology because of the need to have data on premise but not depleting resources and affording high costs for on premise data centers. Data centers now have the ability to work hand in hand with virtual private clouds because of SDN's flexibility allowing for hybrid environments between private and public cloud environments. The public cloud is an avenue to storing data in which can be tied in with internal resources, but with all of these advancements in technology come new attacks vectors for threat actors to attempt at exploiting. The success of these attacks depends on the network and how the attacker initiates the attack sequence [1].

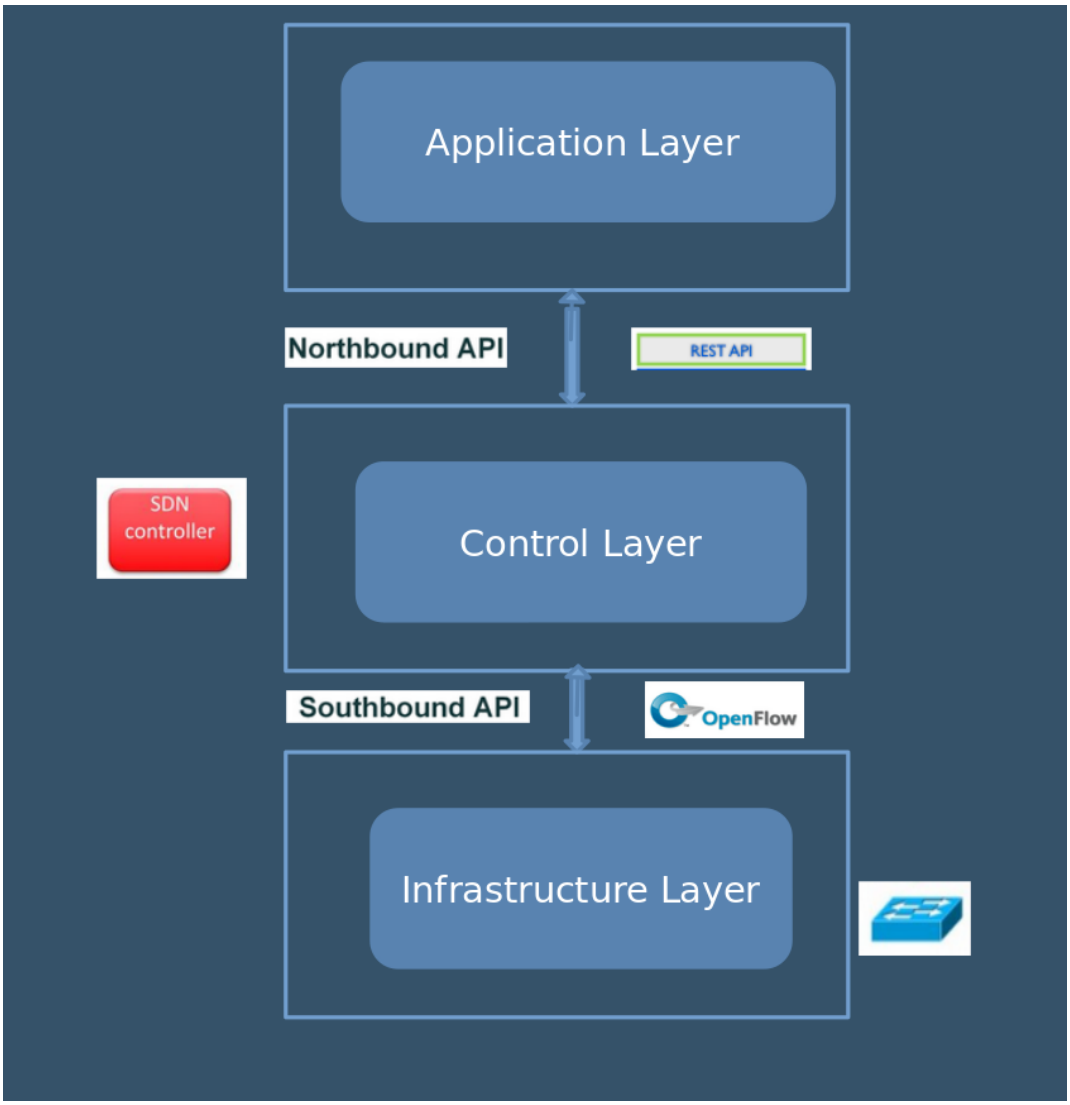
1.2 SDN

SDN has allowed for reduction of complexity in networking by providing a simplified, dynamic and centralized network management system in order to accurately manage the network. SDN does this by decoupling the data plane and control plane of the network in which allows for faster transmission of data as shown in Figure 1.1. The data plane is responsible for forwarding traffic, while the control plane has a centralized controller with a view of the entire network and can be programmed specifically for desired traffic forwarding functionality [2]. On top of the control plane is the management plane where the APIs and logic are written for management of the network. In between the management and control plane is a northbound API, while

in between the control and data plane the southbound API exists. On the northbound API the REST API is a normal standard for communication. On the southbound API, the OpenFlow protocol is the standard for communication. In OpenFlow, data plane devices have flow tables in which are managed by flow rules in the control plane and these rules classify what occurs to traffic. However, if there is no flow rule for a packet, then a request is sent to the controller in order to receive information on what action to perform on the packet. This action normally consists of either dropping or forwarding the traffic. SDN's separation into these three planes is what allows the architecture to be so flexible. It can quickly react to change in the network because the controller sees the network as a whole and not individually making the updates swiftly programmable. The International Data Corporation states that SDN provides security protects against misconfigurations, and they also claim that the SDN market will be worth more than twelve billion dollars in 2022. The importance of SDN in the industry is growing in importance due to the need of having self driving cars, which is intent based networking and supported by SDN.

1.2.1 Cloud's Reliance on SDN

Cloud computing has developed rapidly due to its advantage over traditional computing environments by reducing operational load and cost. Cloud computing requires SDN as the underlying network resource in order for the improvement of cloud manageability and scalability to occur. Cloud computing allows for a high availability environment to be used as needed. The only issues with the development of cloud computing environments are the lag in development of security for them. SDN provides features that can defeat new attacks on cloud computing environments. Denial of Service attacks eliminate the availability in a cloud computing environment. SDN's ability to perform traffic analysis, logical centralized control and forwarding rules makes it easy for the SDN to detect and react to DDoS attacks in a swift manner. However, with all of the separation involved in the different planes of SDN there



(a) The SDN Paradigm

Figure 1.1: The Makeup of a Software Defined Network.

are new attack vectors in which can be used in order to attack SDN from the control plane, data plane or the management plane SDN needs to be properly equipped to defend against these attacks [3].

1.2.2 SDN Platform Threat Vectors

Threat actors have multiple reasons to exploit SDN. The components of SDN and the speed that they provide are created by the controller. The controller is not secure in most scenarios as there are multiple versions of controllers and the support of most controllers has been discontinued. The discontinued support has allowed for bugs to be prevalent in these devices. These controller platforms include but are not limited to the POX, NOX, Floodlight, Beacon, Maestro and OpenDaylight. The controller platform used in this research is the RYU controller in which is supported still while the POX module written in python is no longer supported. Attackers will attempt to compromise the application, control and infrastructure/data link layers. Denial of Service is a common attack, which SDN is vulnerable to due to its protocols such as OpenFlow that can congest the control layer, or all other layers using similar methods to execute.

1.3 Distributed Denial of Service

DDoS attacks are a significant risk to virtualized environments and to the accessbility of assets. In traditional environments, DDoS attacks are attacks performed by targeting a single system from multiple compromised systems called bots or zombies. The attack is performed in order to deplete the resources of the system with the goal of leading the service to be unavailable or incidentally hindered. According to Shi Dong's research there are seven different classes of DDoS attacks. The seven classes include flood attacks, amplification attacks, coremelt attacks, land attacks, TCP SYN attacks, CGI request attacks, and authentication server attacks [4]. For the case of this experiment the crossfire attack will be listed as the eigth due to its complexi-

ties and similarities to a coremelt attack. DDoS attacks have been major threats to individuals in all industries due to the reduction of availability of systems that they cause.

1.3.1 DDoS Variations

A flood attack is categorized by sending a large number of traffic flows to a target system with zombies. The amplification attack assumes that the user's machine has already been taken over and the attacker sends messages from the victim's machine using a broadcast IP address. This then causes the nodes in the subnet to send answers to the victim's system, which causes resource depletion. The coremelt attack is an attack that utilizes a number of machines sending information to one another and has them surge in order to disable a network link, which is similar to the crossfire attack except that there is less flexibility of targeting server areas, the persistence of the attack is lower and that it relies on wanted flows only [5]. The land (local area network denial) attack is an attack that sends a specially crafted TCP SYN attack with the same source port and IP address as the destination. This causes the packet to enter a infinite loop exhausting the system's resources. The TCP SYN attack is an attack on the TCP 3-way Handshake relying on a SYN from the source machine, SYN+ACK bundle from the target machine and an ACK from the source machine to complete the handshake. In this attack, the handshake is left open by continually sending SYN packets in an effort to have the machine waiting on ACK packets to close the connection but never receives one and causes the server to have its CPU time and memory become depleted. CGI server attack is an attack that exploits the vulnerability in common gateway interface systems and sends a large amount of CGI requests in an attempt to consume the CPU cycles of a victim's computer. The seventh type which is an authentication server attack in which an attacker sends a fake signature to the authentication server expending a large number of resources and causing the servers resources to continually lose computing resources with each

signature/attempt [4].

1.3.2 Link Flooding

Detection of the DDoS attacks has become more prevalent in today's society due to the knowledge of the possiblity of the attacks. The commonality of traditional DDoS attacks has left them to be known threat vectors in which controls are primarily focused on defending against. In contrast, indirect DDoS attacks such as link flooding attacks that take servers offline through selective targeting have not had much protection put into place through traditional mitigation methods. In link flooding attacks, the attack is coming from a legitimate source making it difficult to block by IP, signatures or any other method. In addition, detecting link flooding attacks that send low intensity flows at the same time to flood a link can be even more difficult to detect [5]. This problem statement is why this research is focused on cutting the traffic as far close upstream to the source as possible in order to identify and mitigate the most anomalous traffic source. Another emphasis on why we select the crossfire attack is because attackers are becoming more sophisticated with Command and Control (C2) frameworks and DDoS attacks are moving inside the internal network by compromising internal hosts.

1.4 Machine Learning

In machine learning, supervised learning model types are models that are supplied labelled data by users that has the correct answer to a problem in an effort to have the model produce a correct answer on unforeseen future data. The models are built scaled and deployed based on the particular problem so that the problem can be potentially scaled up or scaled down as needed with the algorithm still performing well. Some examples of supervised learning models are logistic regression and random forest. Unsupervised learning in contrast pertains to machine learning models that do not need interaction with the user to learn. These models learn and discover

information on their own and deal with unlabelled data. Unsupervised learning is usually used to solve highly complex problems but are highly unpredictable. Some examples of unsupervised learning are neural networks and cluster analysis. There is semi-supervised learning and reinforcement learning as well, which will be covered in the future work of this paper.

1.4.1 Random Forest

The random forest algorithm is utilized in this research because it is a promising classification algorithm in order to classify traffic as attack traffic based on labelled data. The algorithm will also improve the performance of our network in mitigating the crossfire attack. In this case we know our patterns of data that are most significant for our model to recognize due to the ongoing research of this particular area of SDN so an unsupervised model would not be ideal for this setup. In addition, there needs to be an understanding of what the algorithm is doing and not just performance enhancements from utilizing the algorithm. Logistic regression was not used due to its limitations in handling non-linear relationships in data and multiple expected outputs. Logistic regression was originally chosen as the model for this experiment but it was not a successful implementation for the problem. The random forest model is a compilation of multiple decision trees. Each internal node in a decision tree is essential in testing and branches will represent the outcome of a particular test. Leaf nodes on these trees represent the class labels that were given from the training data. The random forest mitigates the problem of overfitting the decision tree while making the algorithm more accurate when it comes to classification. Consequently, the best split of the decision tree is chosen from a random sample of input variables. Random forest turns out to be the optimal solution for classifying data from all of the input variables in this experiment.

CHAPTER 2: BACKGROUND: THE CROSSFIRE ATTACK

2.1 SDN Awareness

In detection of target area link flooding attacks of SDN, machine learning is highly useful in the classification and mitigation of traffic and it can help the controller train itself to better recognize anomalies [1]. Software Defined Networking has been created in order to increase network scalability and management capabilities to fit the advancement of current network technology. As previously stated, SDN reduces network complexities by providing a dynamic and centralized network management paradigm that separates the control plane logic and the data plane logic in a network [2]. In SDN, the control plane has a centralized, programmable controller that maintains a view of the entire network. In SDN, there is a northbound API and a southbound API, which allow for communication between the layers of SDN. The southbound API is what allows the control plane and the data plane to communicate, by using a protocol like OpenFlow in which allows for communication of SDN switches and the controller. The northbound API is what allows the application plane to communicate with the control plane. The northbound API includes the use of a REST API and others in order to communicate remotely and locally with the controller. The application plane holds unified network monitoring and analysis, network visualization, and security applications including IDS, IPS or a basic firewall [3]. OF devices in the dataplane have flow tables that are managed by the controller that have instructions on how to route traffic called flow rules. All incoming packets are compared with the entries in flow tables and if the traffic type is not found a query is sent to the controller to request further instructions on action requirements for the traffic. The flow rules then receive an update and the traffic is forwarded to the intended destination

or dropped [2]. Traffic will now follow the flow rules with the addition in place so that there is a faster transmission of the packet in comparison to the previous time. This process enforces policy and network configuration requirements in real-time via the controller [6].

2.2 SDN DDoS

There are three different types of DDoS attacks when the target is an SDN. As stated previously there is a management plane also called the application layer, a control plane also known as the control layer and a data plane also known as the infrastructure layer. These three planes are what make up the attack vectors for performing a DDoS attack on a SDN. In the case of this experiment, the main attack is focused on the data layer due to the attack being on the links responsible for forwarding the data. The planes provide an attack vector for those with malicious intent, so discussions of where these attacks occur is a necessary knowledge when dealing with SDN [4].

2.2.1 Application Layer Attacks

The management plane has two different methods by which an attacker can attempt to exploit the layer. The attack could potentially focus on the northbound API or it could focus on attacking the applications running on top to feed the controller. If a threat actor is able to compromise applications involved with the management of the SDN it could cause multiple security issues and concerns for breaches in the SDN.

2.2.2 Control Layer Attacks

The control plane is considered to be the most vulnerable point of the SDN infrastructure due to the possiblity of having a single point of failure in the network [4]. The control plane can be attacked from the southbound API, northbound API or by targeting the controller itself. These issues come into play when the management plane logic causes the controller to make multiple clashing flow rules in which cause

the controller to be more vulnerable because it makes the decisions when queried by the switches of whether to forward the traffic or not. If the switch does not have a flow built into it for some traffic it queries the controller for instructions on what to do with the traffic and these query requests pose a threat vector of DDoS on the controller.

2.2.3 Data Layer Attacks

Data plane attacks can occur through attacking the southbound API or directly attacking the machines in the network. The data plane is normally attacked through flooding the network devices through method of DDoS attacks in order to exhaust the available bandwidth in the topology.

2.3 DDoS Progression

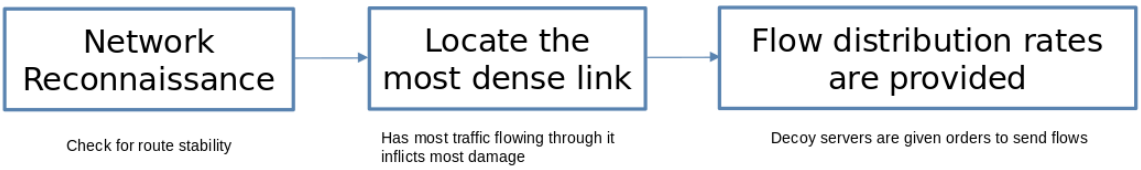
In recent years, DDoS attacks have adapted to become nation-state based attacks. The need to be undetected is prevalent due to all the controls implemented over the years to detect DDoS attacks. The traffic is more difficult to detect in the case of a crossfire attack due to attackers having a link mapped out where bots can send low-rate protocol-conforming traffic towards the target link [7]. DDoS attacks have become far more advanced and are now using a network of bots (botnet) in order to target SDN cloud based targets and environments. A botnet can be defined as a network of computers, which can be controlled remotely that are infected with malware. These bots are then communicating with a bot controller and this is called C2 traffic. C2 traffic allows the botmaster to issue commands to the infected machines. The crossfire attack uses a large-scale botnet in order to execute an attack with legitimate traffic. In a crossfire attack, the machines then direct their low intensity flows to a large number of servers causing an exhaustion of network resources without being able to detect a difference between the legitimate traffic and the malicious traffic [1].

2.3.1 Compromise

Attackers are becoming more sophisticated with their use of C2 servers able to control remote hosts. The complexity that this topic goes into is beyond the scope of this research but it is related. If an attacker is able to compromise an internal host inside the network and is able to control that host without detection then the attacker would have the ability to carry out this attack from the compromised host. The attacker would be able to perform reconnaissance on the network and carry out the basic functions needed to initiate a crossfire attack. With DDoS attacks moving inside networks due to the network security controls implemented outside of the internal network it is now even more prevalent in security to obtain a foothold in the network and be an advanced persistent threat.

2.4 The Crossfire Attack

In the crossfire attack the goal of the attacker is to prevent legitimate traffic from flowing into a specific region of the internet. The two terms that are utilized in order to do this are the "target area" and the "target link". The target area can be defined as the region in which the attacker is trying to cutoff whether it be a city state or even a country. In addition, a target link is an element of a set of network links in which the attacker needs to flood so that the target area is cut off from the rest of the Internet, which makes this the real target [8]. In order to launch a crossfire attack against a target area, an attacker selects a set of public servers within the target area in an effort to construct an attack topology. In addition, the attacker also looks for a set of decoy servers used to create attack flows from the surrounding section of the target area. The attacker then creates a link map from the bot addresses to the public servers in an effort to select the best target links that will immediately cut off the target area's connectivity. After this the client floods the next closest links to the target area in an effort to flood the potential route changes in succession. According



(a) Attacker's Three Steps in Crossfire Attack Following Compromise

Figure 2.1: Attacker's Steps in Crossfire Attack.

to the crossfire attack research, there are three main steps to performing the attack and these consist of link map construction, attack setup and bot coordination [5].

2.4.1 Link Map Construction

In the case of link map construction, often times this is first performed by utilizing network reconnaissance from bots to gather information on other parts of the network. The network reconnaissance allows the bots to find all of the routes to the public servers in the target area and the decoy servers. The bots run these traceroutes multiple times to check for load balancing that could deter an attack as well as to see the stability of the route and if it stays the same or changes. The link map now created needs to be tested for unstable routes. Route instability is caused by load balancing processes and this causes the difference between transient links and persistent links. If a link is persistent then it will always appear on a route, but all transient links which are on the route have to be removed from the set of potential target links [5].

2.4.2 Attack Setup

The next step for the attacker is to find the most flow dense link for the specified target area. This will be a link that has high amounts of traffic transmitted through it and this is how the attacker maps it as a good attack target. The goal is then to find links that can be flooded one after another and inflict the most damage possible to the links. The goal is to be able to flood a few target links and block the majority of connections [5].

2.4.3 Bot Coordination

After the target links are selected the attacker provides each bot the list of decoy servers and the rate at which traffic needs to be sent to the corresponding link. This is the initiation of the attack where the attacker must keep the flow rate of traffic low enough so that it does not trigger the network protection systems, but the second part is that it must be assigned evenly to the bots so that anomalies are much more difficult to detect. The maximum target bandwidth for the target links is already known. After this, the link is exhausted with the attack flows. The difficulty of detection can increase if the attacker dynamically is able to change the sets of target links and therefore making the attack extendable indefinitely. This would be called a rolling attack in which is caused by dynamically enhancing the attack persistence [5].

2.5 Successful Implications

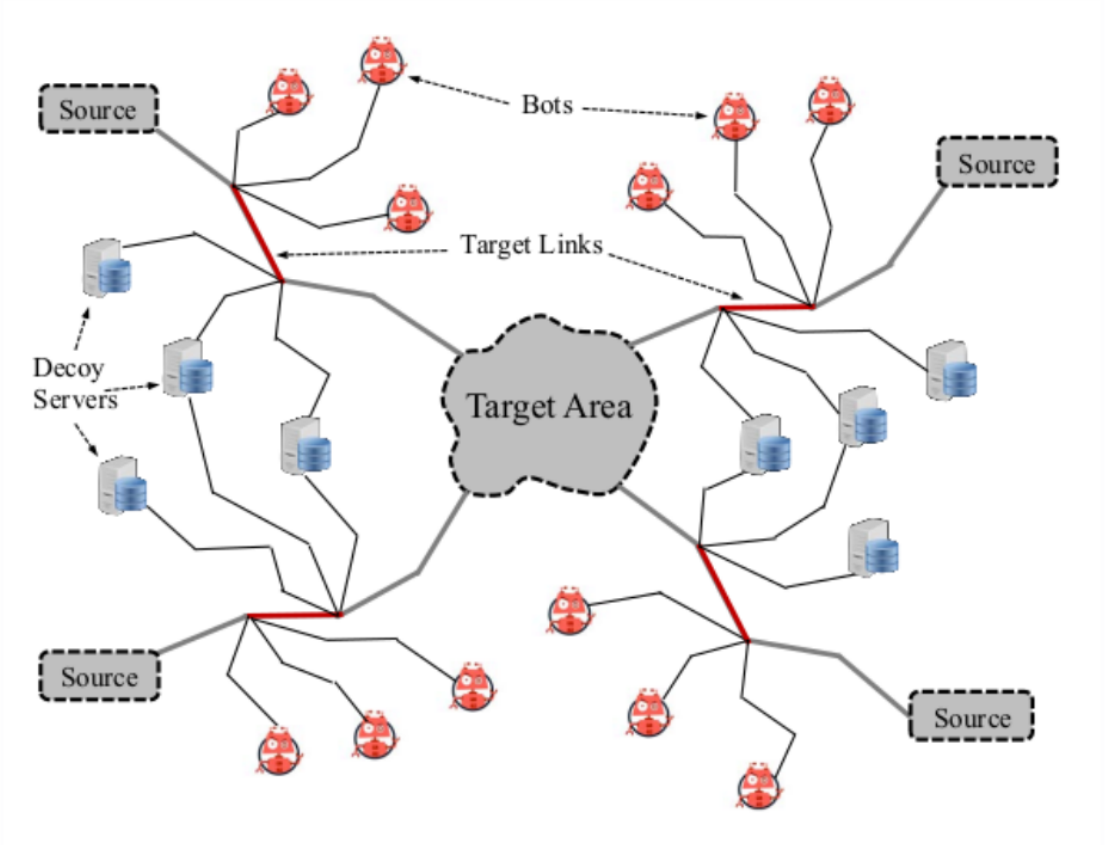
Subsequently, the attack success relies on the method by which the attacker initiates the attack as well as the structure of the network being attacked. The attacker attempts to map the network in order to find a set of target links that are connected to servers of interest. The servers of interest will be determined by the link. Once the target link is determined, the target links are flooded and the legitimate traffic is prevented from reaching the intended destination. In correlation, access for the target area to reach the internet will be denied [1]. A successful attack should be blocking legitimate traffic from entering the target area. Link flooding detection has become more reliant on the SDN paradigm, but the goal of this experiment is to limit the effect of a crossfire attack without causing too much overhead on the controller and avoiding latency when mitigating the attempt.

2.5.1 Sophistication Difference

According to research, DDoS attacks in SDN have been difficult to detect due to the attack traffic characteristics not being easy to identify, the lack of communica-

tion and collaboration between network nodes, the change of the attack tool being strengthened, with the threshold of its use decreasing, the widely used address fraud making it difficult to trace the source of the attack and the duration time of attack being short and response time being limited [9]. The difference in the crossfire attack is that it uses bots with non-spoofed IP addresses to send traffic to the servers available. The bots then send the legitimate low rate traffic to the decoy servers in an attempt to flood the target link. In the case of the crossfire attack, the traffic engineering module is usually the network process utilized to react to network links flooded whether it is malicious or non-malicious. Traffic Engineering calculates the optimal load calculation for each network path while also mapping the traffic flows to a path in which allows each network link to uphold its calculated optimal load [10]. There are multiple methods to identifying the source of the bot traffic in SDN, one of these methods constantly utilized in research is to utilize the traceroute profiles collected from the network by the controller. There is also ICMP monitoring, route mutation and congestion-link monitoring in which will allow for a defender to be able to detect when the attack is occurring and mitigate it [8]. The ideology behind this crossfire attack is to attack on the network level with DNS and TCP traffic while also attacking on the application level by exhausting server resources. After combining the targeted protocol levels we achieve a successful attack vector [3]. A network based mechanism is used in order to defend against the attack vectors. The network based mechanism checks for the highest anomaly in a link by each threshold that the controller has bestowed on a link and is able to track that anomaly back to its original source and not allow traffic from that source in an effort to keep the link bandwidth from becoming depleted [3]. In Rezazad's work, there is an excellent image for visual comprehension of the link flooding attack in further detail as shown in Figure 2.2.

In this research the traffic generating hosts are controlled like a botnet from a command and control perspective.



(a) Link Flooding Attack Visual

Figure 2.2: An Example of a Link Flooding adapted from "Detecting Target-Area Link-Flooding DDoS Attacks using Traffic Analysis and Supervised Learning" M. Rezazad, 2019, Future of Information and Communication Conference, Volume 2. Copyright 2019 by Springer Nature Switzerland AG

CHAPTER 3: RELATED WORK

3.1 Detecting and mitigating link-flooding attacks via SDN

Woodpecker is a defense model where researchers work on the possibility of using SDN's traffic engineering capability to be able to better detect and mitigate a crossfire attack. The woodpecker design research shows that implementing SDN in a regular environment can allow for more flexibility by way of the controller. The controller will upgrade ordinary nodes into SDN-enabled nodes. Once the nodes are upgraded the controller already has flow rules installed in order to load balance the link so that it is detected when a flood attempt occurs. The rules will be triggered in the SDN-enabled switches once the congestion occurs. After this, the controller rules will be enforced to balance the attack flows and this congestion information is constantly updated in the controller's database allowing it to react on the occurrence of an attack. This experiment shows how to develop an optimized SDN solution while upgrading a limited number of nodes. It also how to create a traffic load balancing scheme using SDN [11]. In Linkscope's research the proposed LFA defense system called LFA Defender takes a slightly different approach towards defending against the crossfire attack. Their approach allows for less overhead in the communication channels between control and data plane. The LFA Defender system contains four main modules. Of those four the target link selection and link congestion monitoring are the first two utilized that are designed to detect the LFA. The other two modules are traffic rerouting and malicious traffic blocking in which allowed for the mitigation of the crossfire attack. The target link selection's ability to grab the OF switches flow entries from the controller allows for monitoring agents to be distributed on the links with high flow density. The floodlight controller was used in this environment in

order to assist with the target link selection module written in python. The scheme uses traffic rerouting and blocks malicious traffic in order to defend against the link flooding attack [12]. The scheme is similar to our scheme except that we cut the traffic back upstream to its original source. In the Software-Defined HoneyNet research Kim and Shin use a decoy network in an effort to deceive the attacker and evade an attack of the real network. This scheme leverages global network visibility of SDN in order to create the potential honey nodes which connect the honeynet. This scheme increases the traffic cost by making the attack go through the entire network topology of honey nodes as a first line of defense similar to a honeypot. The scheme needs additional mitigation efforts in order to be considered a mitigation technique [13].

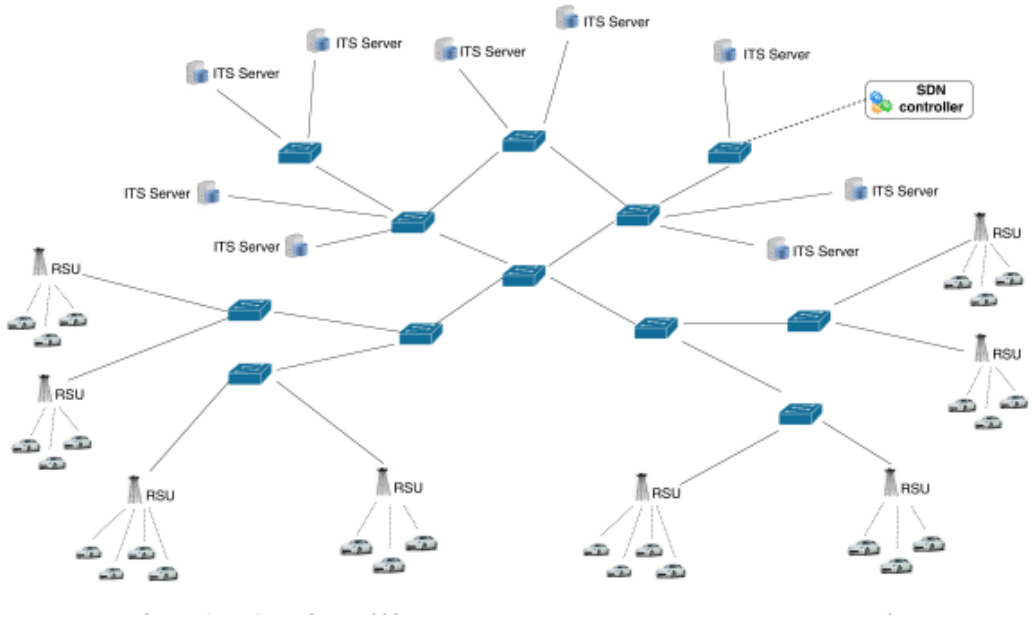
3.2 Crossfire Attack Detection Using Deep Learning in Software Defined ITS Networks

Improvements in intelligent transport systems networks recently have allowed for better safety and security in mobile nodes. In mobile networks, the roadside units and switches are all OF enabled thanks to SDN. Intellignet transport systems have all vehicles on the road communicate using roadside units and the components of ITS networks include internet connected vehicles. As shown in figure 3.1, these nodes communicate in order to predict the future traffic conditions. This paper introduces a unique method in the intrusion prevention of mobile network nodes and detection of security attacks. The researchers developed the deep learning techniques on top of the SDN controller to be run in order to analyze the traffic characteristics on network links. The deep learning techniques based on artificial neural networks can learn traffic behavior to be able to detect whether the network is under an attack or not. The features used in order to create this algorithm were: number of flows, aggregate flow size and timestamp. They developed deep learning techniques from the captured traffic using artificial neural networks, convolutional neural networks, and long short-term machine learning to train the learning model so that it was able

to learn the temporal and spatial correlations among flows that originate from compromised nodes. The performance was evaluated utilizing the mininet-wifi emulation platform showing that the approach performs well in precision, accuracy, recall and F1-score. The results showed that the learning model achieved an accuracy of 80 percent and the LSTM model achieves a detection accuracy of 87 percent. Roadside units allow for the communication between different vehicles to be fluent and possible, but if there is a link flood in which occurs there will be no requests allowed through. Vehicle to vehicle communication can be impacted dramatically making this experiment revolutionary in terms of defending against link flooding attacks in SDN and SD-ITS networks. This experiment relates to the current work because of the logic similarities created on top of the controller in order to mitigate link flooding attacks. The research was interesting in the area of unsupervised learning due to its ability to take unique parts of data that it has never before seen and create a method to solving the problem without human interaction with the data. This is ultimately the future with SDN being a large component in 5G and Vehicular networks, which encompass SDN, CDN, NFV and cloud. This work would be an interesting addition for future research when working with intelligent transport system networks. This would allow for the augment of the research that is currently being performed in order to move forward into a interesting area of 5G research. [14].

3.3 DDoS Attack Detection based on SVM

In Ye's work, the experiment utilizes five nodes in order to setup the SDN environment all on separate connections. The first two machines in the topology are used as bot hosts and the next two are used for generating normal network traffic samples. The user attempts to utilize the customizability of the packet generator, hping3, and its TCL language in order to send and receive data packets by describing a packet's string or binary representation. In the experiment the sample period for learning is in three second intervals. During this process, the flow table data of sixty periods is



(a) An Example of an ITS Network

Figure 3.1: An Example of an ITS Network. From Crossfire Attack Detection Using Deep Learning in Software Defined ITS Networks. Narayanadoss 2019, 89th Vehicular Technology Conference.

collected in the openflow switch. This data is then normalized and processed and the normal samples as well as the DDoS attack flow samples are shown in order to show the number of flow entries increasing dramatically in the event of an attack. The attack is performed based on pseudosource random IP addresses and port numbers and this causes the destination host not to be able to respond in a timely manner. This causes the proportion of interactive flows to decrease dramatically, in contrast to the normal circumstance of interactive flow entries being large and fluctuating in the normal range of traffic. In this experiment, Support Vector Machine (SVM) machine learning functions were used in R studio in order to train the data to get the SVM model and predict the test data by using the model. The client then uses a formula in order to calculate the detection rate and the false alarm rate of the model in order to understand the accuracy in checking for an attack. The legitimate traffic generated in the experiment is TCP, UDP and ICMP traffic and the attack traffic consists of these

three types of traffic as well. The researchers conclude that the model is 95.24 percent accurate and achieves a false alarm rate of 1.26 percent accroding to the model that was used in the experiment. The model had a low rate of ICMP flow detection but this was concluded to be on behalf of the information that ICMP packets lack. ICMP traffic has no source or destination port making it harder to check by the use of the researchers model. Overall, the experiment was a success in terms of the researchers concluding remarks [9].

3.4 Evaluating Link Latency in Distributed SDN Based Control Plane Architecture

In this work it goes through the importance of having a monitoring system for link latency due to it being a crucial feature in the transport infrastructure of a 5G service. It proposes the automatic monitoring of link latency in an SDN control plane without introducing overhead in the southbound interface. In this research they leverage the exchange of the link layer discovery protocol, which is utilized by the controller discovers the links in a network topology. They utilize the openflow packet out messages usually sent by the controller to manage switches and the packet in messages sent back to the controller for the initial setup. It utilizes this communication method to send an echo request and echo reply containing the time stamp allowing for the controller to be able to evaluate the round trip time of the openflow channel. The methodology also uses the offset evaluation between two controller instances with the exchange of ad-hoc packets. This allows for the clock offset between those different controller instances to be monitored. In the results of this work it reports on the link latency measured maximum and minimum and allows for the measured latency to be acknowledted in terms of milliseconds. The errors present in the maximum and minimum values were related to the desynchronization of processes during evaluation such as link layer discovery protocol, round trip time computation and offset evaluation. The synchronization of these processes would allow for more exact results. Overall,

this work was a success in the process of monitoring links but poses no details on mitigating a link that has been identified with high latency [15].

CHAPTER 4: METHODOLOGY

In this research there is a need for a model that can accurately classify when traffic is normal, before an attack is about to occur and when an attack is occurring. Dropping traffic from the ethernet interface that is generating the most anomalous traffic in terms of a spike will allow for mitigation closer to the source and less impact on the amount of users not able to utilize the link being targeted. We utilize flow, port and group statistics in order to feed data into our classification model. Once that data is fed into the classification model it will then make a decision on how the traffic will be classified. The algorithm evaluates if the link needs to have traffic flow through it or if that interface needs to be closely monitored. If it needs to be closely monitored then the traffic is classified as forthcoming attack and is on the verge of becoming attack traffic, which is an unusual anomaly. Attack traffic needs to have all traffic dropped on that interface. As more interfaces become anomalous in their generation of traffic then more interfaces will be cut off from being able to deliver traffic to the intended destination. Therefore this will free up the bandwidth utilized by links in the network. This will also allow for higher throughput and bandwidth of links in the topology.

4.1 Flows

What are flows? Flows are entries in which provide instructions on the handling of a packet in SDN. The SDN controller constantly updates the flow entries in the flow tables in order to have more instructions in the data plane on how to forward traffic. Each SDN switch contains flow tables and a set of flow entries inside those tables. The flow entries have match fields, counters and instructions to apply to packets in

which match a particular entry. If there is no entry in the data plane the control plane is queried for further instruction.

4.1.1 Exploiting Flows

In the link flooding attack, the attacker is able to flood these links in the topology abusing the flows that are currently built into the controller. If the switches know to always forward traffic of a particular type from a certain source to a destination then an attacker can find out the available bandwidth on the link and attempt depleting the amount of bandwidth available in that link. If there is no rule to drop all traffic from potentially anomalistic traffic generation source then the attacker will succeed in not allowing traffic to reach its particular destination. If we are able to create a monitoring engine for the traffic in the network then we will be able to create flow rules that are sophisticated enough to drop traffic at the interface source.

4.2 Monitoring

In order to make an accurate classification of when an attack is about to happen versus when it is normal or under attack there is a need for monitoring of the network components. The monitoring will need to be performed on the flow, port, link, path and switch levels of the network [16]. Monitoring at these levels will allow our algorithm to make an accurate decision when dealing with a particular interface generating an abnormal amount of traffic. Performing monitoring of the topology at these levels will allow for an ample amount of data to be generated. The more data that is available to feed into our model for analysis then the classification accuracy will increase.

4.2.1 Mitigation

As our monitored data is fed in from the controller to our supervised learning model it will allow the model to make decisions on whether to drop all traffic on a particular interface. The optimal solution to remediation of a denial of service attack is limiting

the amount of resources and people that are affected. If a design is able to mitigate the attack upstream from the original generation of the traffic, then the attack would be ineffective as the source of the attack has been neutralized.

4.3 Classification

In this research, random forest was utilized for the exceptional classification ability of the algorithm. Random forest is a combination of multiple decision trees. Decision trees are a form of supervised learning that utilize a created training model in order to predict the value of importance, which is also known as the target variable. In decision trees, prediction of the labelled data begins from the root of the tree and splits into two or more sub-nodes in order to find the corresponding value. In decision trees each internal node maps to a feature, the branch matches a decision and the leaves represent the label chosen. In this experiment the random forest of decision trees was used to classify data based off what label the data most closely resembled. Classification is a process that utilizes learning and predicting. In this research the input variables fed into the algorithm allow for it to identify the optimal classification for the interfaces resembling an anomaly.

4.3.1 Why Random Forest?

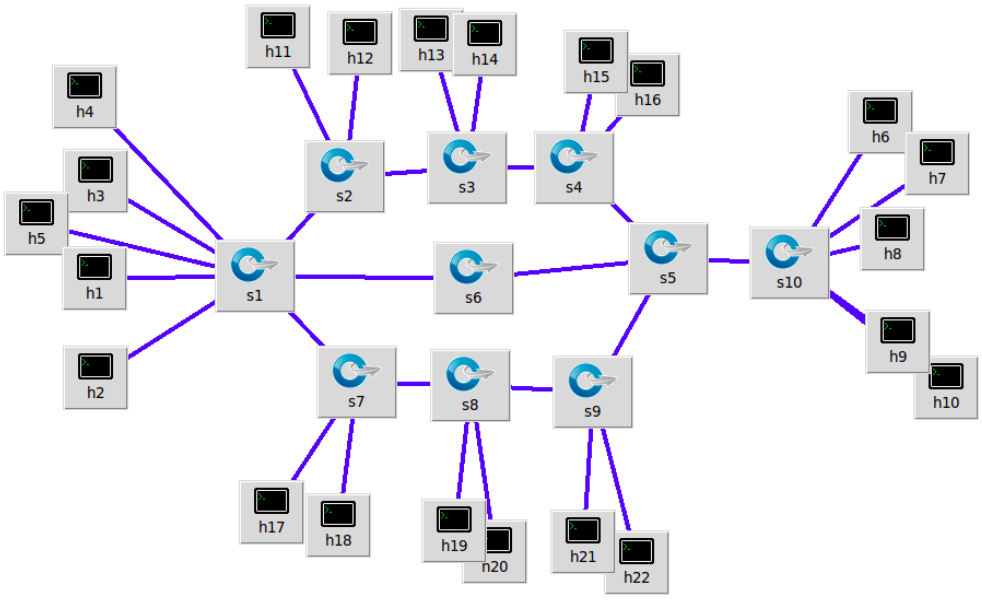
Decision trees are prone to overfitting due to them not being able to fit all samples in the data that are supplied into the tree. This therefore allows the training dataset to classify at a great percentage but the actual test set of data will not perform accurately. In random forest there is an ensemble of decision trees allowing for all of the models to operate in accordance with one another. This allows for more features to be captured in each tree, thus making the model more accurate in solving the problem. In this model the individual trees' predictions have low correlation with each other allowing for a diversity of decision making and no compilation of error on the same mistake in classification. The principle utilized is bagging, which allows for

individual trees to be able to randomly sample from the dataset resulting in different trees. Based on the predefined labels for the data the interfaces are classified based off of normal traffic, forthcoming attack traffic and attack traffic. The model is able to accurately classify links in the topology to mitigate the effects of the crossfire attack due to the model's efficiency in the process of classification.

CHAPTER 5: EXPERIMENT SETUP AND RESULTS

5.1 Experiment Setup

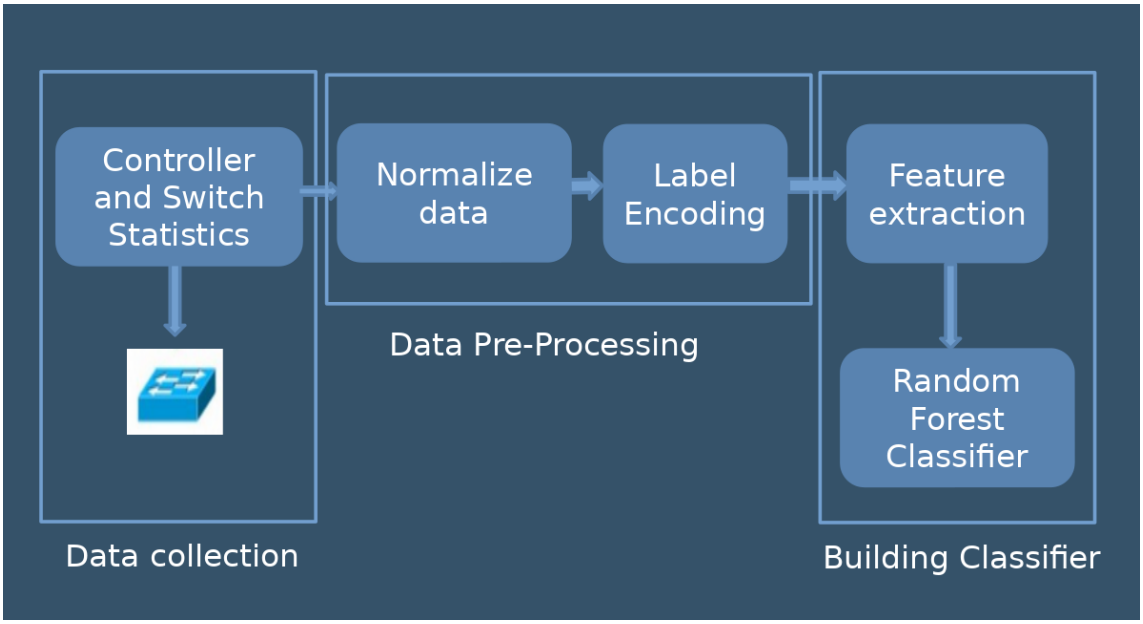
In this experiment the ryu controller was used in accordance with openvswitch OF enabled switches in which were deployed in a ubuntu environment. The topologies utilized in the experiment were generated using mininet. The ryu controller module utilized in this experiment is a slight modification of the simple switch stp 13 available through the ryu controller install packages. The topology contains loops so this module works with the spanning tree protocol and allows for the traffic to be able to communicate utilizing bridge protocol data unit. There have been multiple sources of the traffic created and in this experiment, the D-ITG traffic generator and ostinato tool were utilized to simulate background traffic. In addition, a couple scripts were written to automate the D-ITG creation process as well as the ostinato generation process. Other scripts in the background were also running to make the environment comparable to enterprise environment traffic generation, due to the limitations on the amount of traffic that D-ITG can generate. The SDN switches are used to output flow statistics on the traffic to the link targeted. The controller also has statistics that it will report to the machine learning model. The topology represents an environment where all traffic transmission is attempting to access one particular segment of the topology. This segmented region has one link in which can be flooded and stop the entire flow of traffic to the zone. This link will be recognized as the target link due to this zone's hosting all of the legitimate servers and hosts. For the sake of computing resources the link has 50Mbps allocated and available for traffic transmission flows and after this bandwidth is exhausted the link will no longer allow traffic through. Compromised nodes and decoy servers are simulated in the network coinciding with



(a) An Example of a Network Setting Utilized for Testing Scalability

Figure 5.1: An Example of One of the Topologies Used in This Research.

regular hosts as shown in figure 5.1. The hosts generate normal traffic throughout the duration of the experiment and the slight traffic spike is generated from these same hosts making the anomaly detection and restriction more difficult to locate. In this experiment, the traffic is generated in multiple different classifications including: DNS, VOIP or TCP, UDP, DCCP, ICMP and SCTP. The D-ITG traffic generator allows for the diversifying of traffic generation as well as ostinato. The goal is to have the controller be able to cut traffic flow of an interface at the source of the abnormality upstream in order to limit the amount of legitimate traffic not allowed to travel to its destination. This will therefore minimize the amount of users affected. The tools utilized in this experiment include: python, mininet, the RYU STP controller, openvswitch switches , ostinato and D-ITG. The machine learning model is ran at the same time as the controller to collect statistics, classify and send instructions to the controller for action on classified link flooding attack traffic.

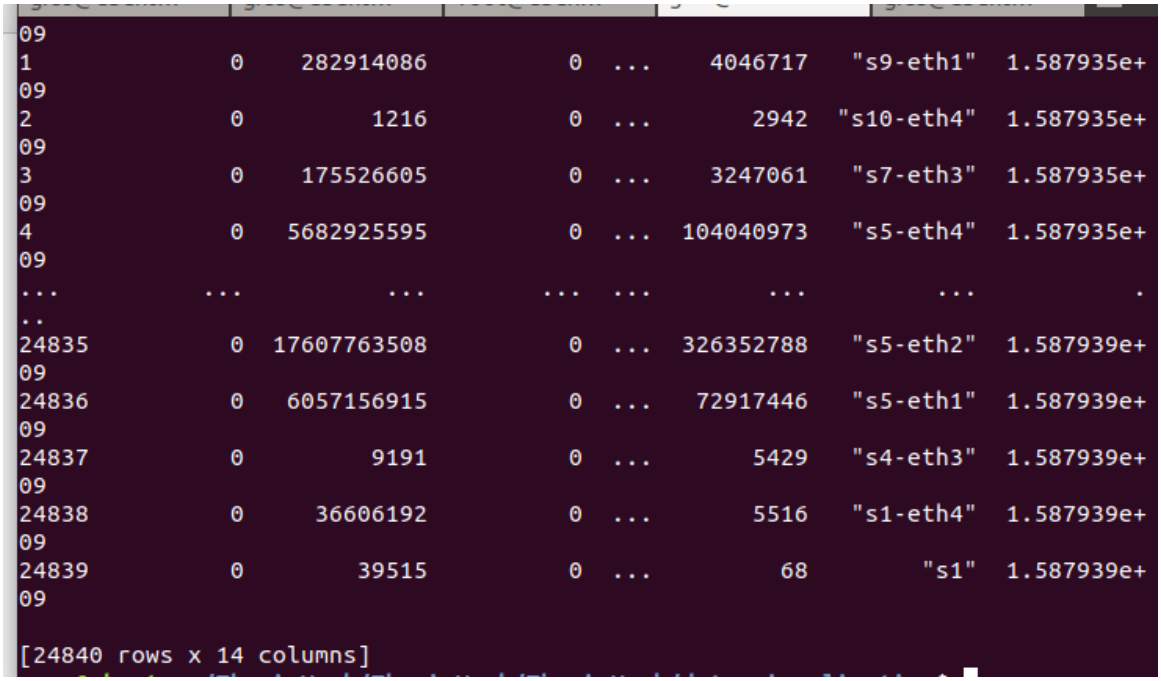


(a) The Model Utilized for Classification

Figure 5.2: The Model Design for the utilization of Random Forest.

5.1.1 Crossfire Defense

The crossfire attack is difficult to detect because of its spike in the same traffic that is legitimate. The user will need to implement the random forest classifier model with an n estimator value of 500 at each monitoring level. This will allow the model to ingest the data needed for it to perform at its optimal level. The user needs to then create a message that will inform the controller on the status of links in the network. The data needs to be cleaned, the accuracy needs to be checked, and there needs to be constant controller to openvswitch statistic requests. Once this data is all gathered the actions for links that are classified will be communicated to the controller for action to be taken. The amount of time it takes to flood the link from the multiple different sources is what allows for detection and mitigation inside the SDN. Once the interface is identified that is reaching an abnormal threshold it is classified as about to be an attack. Once this is classified accurately the controller is able to accurately assess what to do with future traffic if it continues to develop into a level, which insinuates attack traffic. The added intelligence from the model communicates the



```
09
1              0    282914086          0  ...    4046717   "s9-eth1"  1.587935e+
09
2              0         1216          0  ...       2942  "s10-eth4"  1.587935e+
09
3              0    175526605          0  ...    3247061   "s7-eth3"  1.587935e+
09
4              0   5682925595          0  ...  104040973   "s5-eth4"  1.587935e+
09
...          ...          ...        ...  ...        ...         ...           .
..
24835          0  17607763508          0  ...  326352788   "s5-eth2"  1.587939e+
09
24836          0   6057156915          0  ...   72917446   "s5-eth1"  1.587939e+
09
24837          0         9191          0  ...       5429   "s4-eth3"  1.587939e+
09
24838          0     36606192          0  ...       5516   "s1-eth4"  1.587939e+
09
24839          0        39515          0  ...         68        "s1"  1.587939e+
09

[24840 rows x 14 columns]
```

(a) Raw Data Collected

Figure 5.3: Raw Collected Data Example.

instructions so that it locates the source with the highest variation off of normality first and then drops all traffic off of that interface closest to the source upstream. The controller for when there is an unusual spike of traffic from one segment of the network RYU has function that is able to set the traffic to disable communication using the REST API also known to be part of the management plane. Through this implementation, the crossfire attack on the targeted link is able to be mitigated at its source because the controller is able to act on data fed to it by the machine learning model.

5.2 Data Collection and Results

In terms of data collection statistics are collected from the controller and open-vswitch statistics. The port, flow and group statistics of the network were monitored by the controller meanwhile the switches also contained volatile information that was captured. The experiment was ran for multiple hours in order to collect data on the regular state of the network and the attack state of the network. A python script

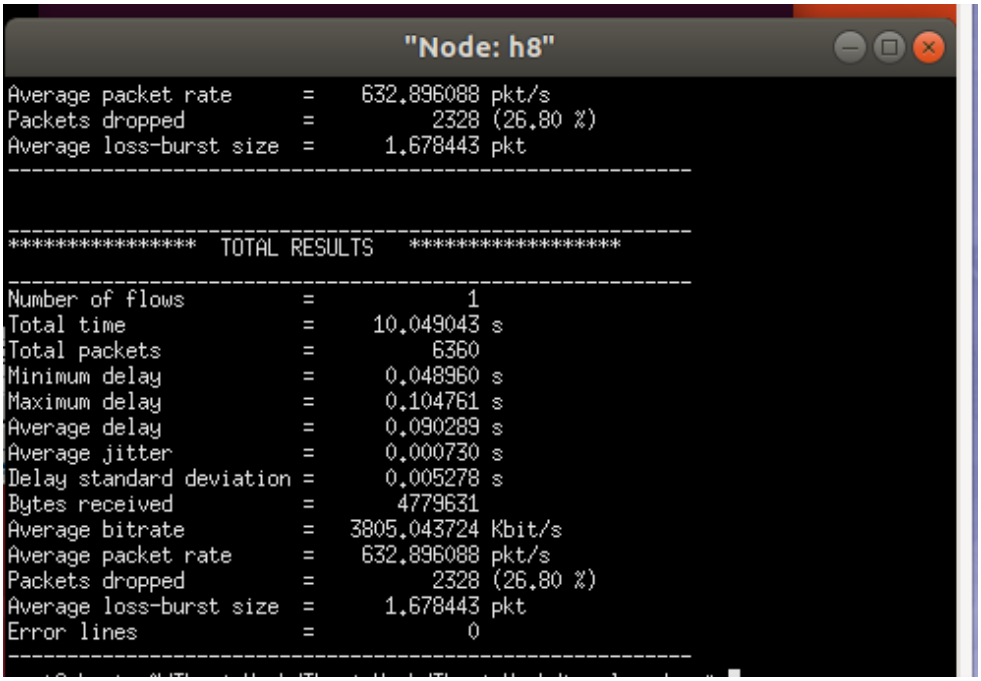
was created to continually extract the data from the controller and switch statistics to write that data to a file. This data had to be formatted and normalized once collected. The normalizing processing is done with mmscaler used in sklearn because it allows for unit variance allowing the labels that are already created to be efficient. The data supplied to the algorithm included the MAC address, interface, transmitted and received packets, the drop rate, destination, port details and the timestamp. The data was labeled in three different ways as normal, about to be an attack or attack traffic. The labeled data allows for an accurate method to be able to check the accuracy of the model. The feature extraction was performed for the data provided and then fed into the Random Forest Classifier. A validation curve was utilized that computed the validation score on multiple validation sets of data in order to tune the hyperparameters of the model. Of those hyperparameters that were tuned in the model the most important was the augment to an n estimator value of 500. The default value for n estimator is 10 but this only allows for ten decision trees to be utilized in the model causing problems with overfitting of the data. A validation score was generated using k-fold cross validation utilizing with k's value set to 10. This data was then used to calculate the minimum average that had a percentage of 90.31 percent. The accuracy was checked on the experiment and it achieved an accuracy of 90.63 percent.

Table 5.1: Confusion Matrix

Value	**Forthcoming Attack**	**Normal**	**Attack**
α	β	γ	δ
Value Type	Prediction	Prediction	Prediction
True Values	6	0	1
True Values	1	17	0
True Values	1	0	6

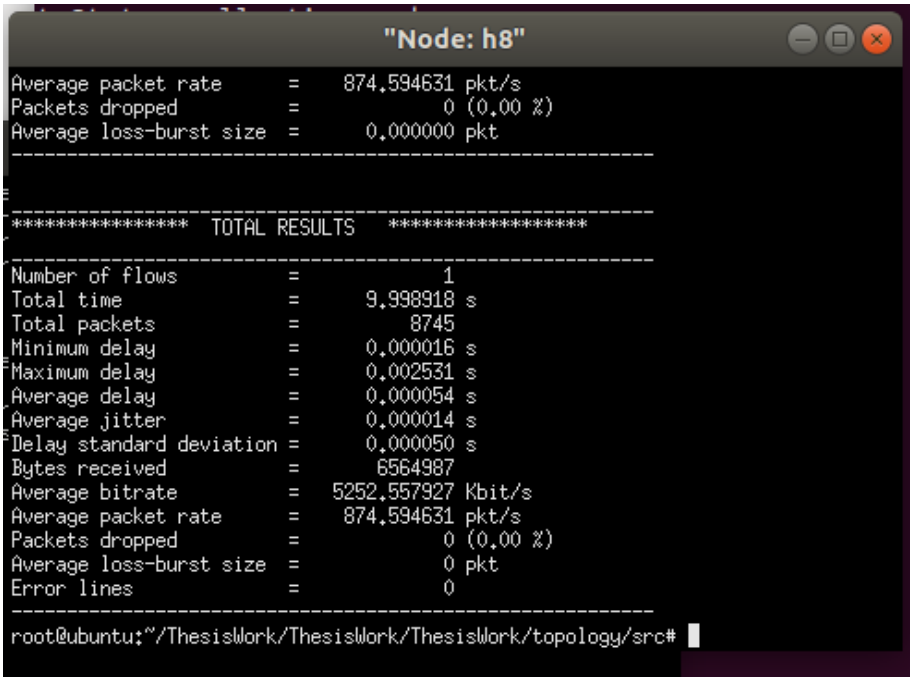
5.2.1 Effects on The Network

The load balancing of the controller without the added intelligence of the machine learning model is not able to maintain the amount of traffic flows in the network towards the important servers behind the target link. This causes an increased amount in the packets dropped and the delay in transmission of packets to that region of the network. The bandwidth of the target link was periodically checked with a UDP sent from a dedicated host checking the status. As a result of the attack on the original topology with no additional intelligence 25 percent of the UDP packets transmitted are dropped due to bandwidth exhaustion on the link. The machine learning model was able to allow more flows in to the link over time and increase the throughput allowing all of the UDP flows to be transmitted and received successfully through the link with no packets dropped. The delay in transmission increased for the UDP flows by 400 percent in the original topology without machine learning in comparison to the topology working with the machine learning model. Therefore our model was able to accurately mitigate a crossfire attack from occurring although it was strict in its implementation. As our model learned the network topology more it began to cut more links inside the topology allowing for higher network throughput for our links but it was cutting some legitimate traffic as well. The algorithm was a success in total for the fact that it was able to solve our problem of creating a method for mitigating a crossfire attack in SDN.



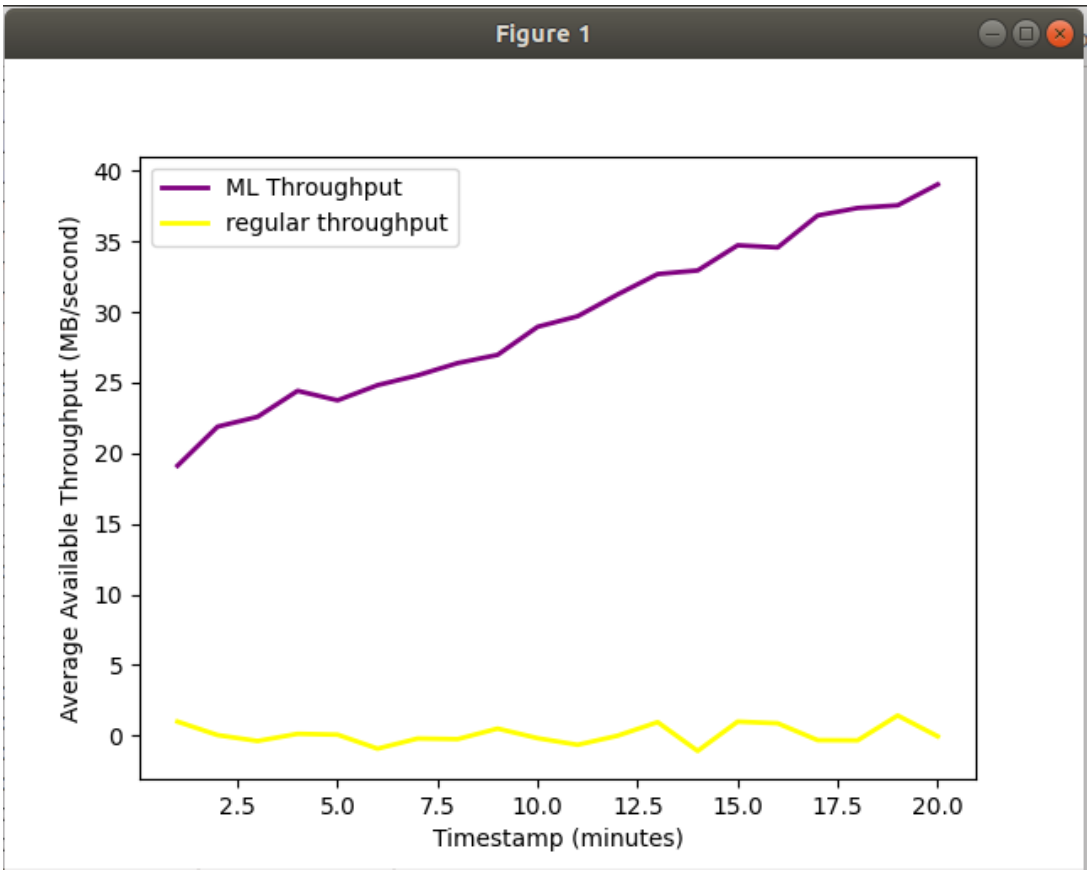
(a) UDP Flow to Check Statistics On the Target Link for the Topology with no Model

Figure 5.4: Statistics on The Target Link with no Machine Learning.



(a) UDP Flow to Check Statistics On the Target Link for the Topology Enhanced with the Model

Figure 5.5: Statistics on The Target Link with Machine Learning.



(a) Target Link Throughput over time. ML Model Added Intelligence vs. Original Topology with no additional Intelligence

Figure 5.6: Target Link Throughput Over Time.

CHAPTER 6: CONCLUSION

The crossfire attack is a nearly undetectable and indistinguishable target area link flooding attack in which exhausts the bandwidth of network links. In a traditional DDoS attack flood end nodes, while in contrast the link flooding attack attempts to flood the intermediate links [13]. The crossfire attack utilizes a large scale botnet in order to initiate legitimate traffic flows for saturation of the links bandwidth in the network. Once the bandwidth of a link is depleted, the link can no longer accept any types of requests causing multiple users requests to not be completed and their traffic dropped. This denial of availability is what makes the crossfire attack important in regards to detection and mitigation. In this paper, a mitigation effort in regards to the crossfire attack is proposed and defended. The detection and mitigation ability of the network is built into the logic of the machine learning model in which augments the controller logic. The algorithm was accurately able to classify all interfaces with a test score of 90.63 percent. The available network bandwidth able to be maintained during the attack after the model reached its maximum bandwidth allocated for the link was 50MB and was able to be maintained at 35MB as traffic generation continued to increase due to the model's precision. In conclusion, the model allowed for less delay in the transmission of data, a lower amount of packets dropped over time to our target link, and a higher average packet rate.

6.1 Pros and Cons

The model was efficient in mitigated a crossfire attack from being able to occur on an important link in the topology. This implementation works cutting traffic closest to the source in which limits the amount of users affected by a link flooding attack.

The model does not cause a lot of unnecessary overhead to occur in the controller due to the implemented design that was utilized.

CHAPTER 7: FUTURE WORK

In the future, we would want to scale this on a 5G implementation level. In addition, we would want to build network slicing on top of the topology that will provide isolation between slices and allocate bandwidth to machines with higher traffic output from areas of lower bandwidth consumption in the network [17]. In order to do this we would utilize a generative adversarial network, which is semi supervised learning letting the algorithm learn how to perform network slicing on the network learning it as if it were a game. Additionally, we would want to utilize a solution that allows for the learning the network as it is continually being augmented. This would allow for the movement of the work into real time optimization of load balancing and routing in addition to the solution proposed in this paper [18]. In addition, there would be a desire to utilize reinforcement learning in order to better optimize the load balancing process inside the environment. Arena would also be looked into as a safe and quick method to adjusting the algorithm and finding a solution to the problem working with the algorithm step by step. We would figure out how this implementation works at scale. Lastly, we would make service agnostic requests to a web server and dns server that we spin up in order to check the actual effects of this experiment in an enterprise like environment with web servers, dns servers and mail servers etc. This will allow for the understanding of an optimal solution for this attack moving forward in a 5G ecosystem.

7.1 Problem Statement

Can we raise an early alarm for crossfire attacks when network traffic demonstrates an anomaly? Can we identify and mitigate the upstream traffic generation closer to

the source? In this research we were able to create a successful method to slow an attacker from flooding a volatile link in the topology. We utilized the random forest machine learning algorithm in order to be able to classify when network traffic was demonstrating anomalistic behavior. We were also able to utilize the compilation of the algorithm and the controller in order to mitigate the upstream traffic generation closest to the source. This model was able to operate at 90.63 percent efficiency in classifying all traffic. The future model will also include reinforcement learning for load balancing as well as semi supervised learning in order to perform network slicing of the topology.

REFERENCES

[1] M. Rezazad, M. R. Brust, M. Akbari, P. Bouvry, and N.-M. Cheung, "Detecting target-area link-flooding DDoS attacks using traffic analysis and supervised learning," vol. 887, pp. 180–202.

[2] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," vol. 7, pp. 34885–34899.

[3] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," vol. 18, no. 1, pp. 602–622.

[4] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments," vol. 7, pp. 80813–80828.

[5] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *2013 IEEE Symposium on Security and Privacy (SP) Conference*, pp. 127–141, IEEE.

[6] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, pp. 184–189, IEEE.

[7] L. Xue, X. Luo, E. W. W. Chan, and X. Zhan, "Towards detecting target link flooding attack," p. 17.

[8] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using SDN-based moving target defense," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 627–630, IEEE.

[9] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," vol. 2018, pp. 1–8.

[10] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE.

[11] L. Wang, Q. Li, Y. Jiang, X. Jia, and J. Wu, "Woodpecker: Detecting and mitigating link-flooding attacks via SDN," vol. 147, pp. 1–13.

[12] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and mitigating target link-flooding attacks using SDN," vol. 16, no. 6, pp. 944–956.

[13] J. Kim and S. Shin, "Software-defined HoneyNet: Towards mitigating link flooding attacks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 99–100, IEEE.

[14] A. R. Narayanadoss, T. Truong-Huu, P. M. Mohan, and M. Gurusamy, "Crossfire attack detection using deep learning in software defined ITS networks," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–6, IEEE.

[15] A. Sgambelluri, X. C. Moreno, S. Spadaro, and P. Monti, "Evaluating Link Latency in Distributed SDN-Based Control Plane Architectures," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, (Shanghai, China), pp. 1–6, IEEE, May 2019.

[16] W. Queiroz, M. A. Capretz, and M. Dantas, "An approach for SDN traffic monitoring based on big data techniques," *Journal of Network and Computer Applications*, vol. 131, pp. 28–39, Apr. 2019.

[17] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, and P. Kankariya, "Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, (Las Vegas, NV, USA), pp. 0852–0857, IEEE, Jan. 2020.

[18] K. Sawada, D. Kotani, and Y. Okabe, "Network Routing Optimization Based on Machine Learning Using Graph Networks Robust against Topology Change," in *2020 International Conference on Information Networking (ICOIN)*, (Barcelona, Spain), pp. 608–615, IEEE, Jan. 2020.