Rotating Adaptive Network Defense (RAND)

Design Document

Team ID: sddec18-07

Client: Argonne National Laboratory, Dr. Benjamin Blakely and Joshua Lyle

Faculty Advisor: Dr. Hongwei Zhang

Team Members and Roles:
Andrew Thai — *Project Manager*
Connor Ruggles — *Usability Manager*
Emily Anderson — *Deliverable Manager*
Ryan Lawrence — *Communication Manager*
Corey Wright — *Quality Assurance Manager*

Team email: sddec18-07@iastate.edu
Team Website: https://sddec18-07.sd.ece.iastate.edu/

Table of Contents

List of Figures

List of Definitions and Acronyms

SDN: Software-Defined Network

- Software-defined networking is a concept where the actual routing of data packets is moved to a separate layer and is taken care of programmatically by a network controller, that then sends the packets down to the main network switch to route to the individual servers on the network.

MTD: Moving Target Defense

- This is a concept where you detect if a specific machine is being attacked and you have preset rules to mitigate to rotate that machine out of being public facing, and rotate in a "honeypot", or something that looks like a real machine but it distracts the attacker long enough to block them out.

NIC: Network Interface Card

- This is the physical device that connects all of the machines connected to a switch, to the Internet.

CDC: Cyber Defense Competition

- A type of competition where teams try and defend a set of servers against a team of attackers in a pre-defined scenario.

VM: Virtual Machine

- A software emulation of physical aspects needed to run a full computer operating system.

Honeypot Server

- A server that is used to trap hackers from accessing the actual production systems.

IDS: Intrusion Detection System

- A system that monitors a network for malicious activity.

1 Introduction

1.1 Acknowledgement

Our clients, Dr. Benjamin Blakely and Joshua Lyle from Argonne National Laboratory, have been the biggest contributors to our project. They have given us direction pretty much every step of the way and helped us solve any problems we came across. Our faculty advisor, Dr. Hongwei Zhang, has also helped us throughout this process on larger deliverables such as our presentations, design doc, etc.

1.2 Problem and Project Statement

As the pace of advancement in information and operational technology systems rapidly increases, cyber-attacks become more sophisticated due to the additional resources available to cyber criminals. They have become harder to detect and more effective at penetrating networks. Cyber criminals might spend weeks and months gathering information on target networks to plan out their attack, making sure that they have the right information so that their attacks will work efficiently and effectively.

Our solution consists of a software defined network (SDN) controller that dynamically adjusts where incoming packets are directed when they are being transmitted to a server. By doing so, we will be able to route traffic on the fly to migrate, take down, or add new servers to the network without any downtime. We will utilize a SDN as a moving target defense (MTD) system. The controller we develop will dynamically configure the network to detect any packets that are malicious or come from an information gathering reconnaissance tool and direct them to dummy servers, also known as honeypots. This will prevent or delay an attacker from obtaining any reliable information about the network. This could result in many wasted attempts to grab information regarding the constantly changing network, thus allowing the network to be more difficult to attack than a static configuration.

1.3 Operational Environment

This design will be used in a location where public-facing servers are located. For example, many institutions use a demilitarized zone (DMZ) network segment for web servers or email services which require incoming requests to be served. Such servers could be located in datacenters or on-premises at a facility belonging to the owner. Any physical hardware, such as switches or a server to host the controller, that would be put into place would be able to withstand standard networking environments such as networking closets or datacenter cabinets.

1.4 Intended Users and Uses

The intended users for the developed product are any company with services that use multiple virtual or physical servers, whether internal or not, such as hosting a website or any other service that uses some sort of a network connection between multiple other servers. This design can also be used for government or military institutions to protect from various information gathering attacks.

This SDN MTD product will provide an extra layer of security by dynamically routing traffic to an array of systems thus allowing for a wide variety of maneuvering to impede network scanning.

1.5 Assumptions and Limitations

Assumptions:

1. Physical or virtual switches used must support the OpenFlow protocol.
2. All switches must have a route to connect to the SDN controller.
3. There are companies/customers willing to implement SDN on their own network.

Limitations:

1. Not ideal for a home network. The resources required and scope of the whole system would be inefficient for the size of a typical home network.
2. Load balanced servers must be in same geographic location.
3. Snort logging takes up a very large amount of space.

1.6 Expected End Product and Deliverables

The end product will consist of :

- Research data including but not limited to:
    - Background on SDN and the protocols selected for our implementation (e.g., OpenFlow);
    - Gaps in existing implementations that are similar to what we developed;
    - The threat model defining the scope of attackers the product is designed to defend against;
    - Details and reasoning behind our implementation of the SDN MTD product (including diagrams, when appropriate);
    - The evaluation methodology used to assess the performance of the product and degree to which it counters the in-scope attacks;
    - Results of the assessment;
    - Recommendations for future work;
- Source code or configurations for a SDN controller with basic routing rules (and documentation for creating more specific rules);
- An instruction process to aid the end user in deploying the controller onto a virtual machine network; and
- Any other configuration files needed for the system to work as expected.

Research Data - This is the primary deliverable and will lay out the procedures for the entire project so that the methodology can be assessed, replicated, and extended.

Install Directions - Detailed instructions on how to manually set up the controller and other machines necessary to the system, like the Snort machine.

Configuration Files - Any configuration files needed for the Floodlight controller setup or Snort setup

Other deliverables include usability and effectiveness of the system which show tested results that describe the impact of using this system as well as if it actually makes a significant difference than just using a regular network.

2 Specifications and Analysis

2.1 Proposed Design

We revised our project design last semester into this improved design to help create a software defined network moving target defense system. This design consist of two servers: Floodlight Controller and Snort. The Floodlight Controller is our brains of the system in which it handles all the traffic routing and dynamic packet flow. The Open vSwitches that are within the network will connect to the Floodlight Controller and will route traffic based on the rules that are created within the Floodlight Controller. The Snort server is our network-based intrusion detection system which handles network traffic and alerts us of any malicious traffic that may going through our network. These two systems will be put in front of the network of our Web Server allow for traffic to be blocked or redirected to a honeypot server.

In our design, we have incoming traffic coming from the Internet, once the traffic makes it into the network where the Snort server is located, Snort will start analyzing all the traffic and alerting for any malicious traffic that is incoming based on the community rules and the local rules that we have created. We decided to use Snort as our intrusion detection tool because it will be scalable in the long run with updated community rules. Once Snort creates an alert we have a custom python script that we created and runs every 30 seconds on the system (based on using the system's crontab) that checks if there are any alerts in the file /var/log/snort/alert. If there are any logs, the script will parse through the alerts to grab the source and destination IP. Once the script gathers the IP it creates a flow to either block or redirect the current traffic to a honeypot server by communicating with the Floodlight Controller via API calls.

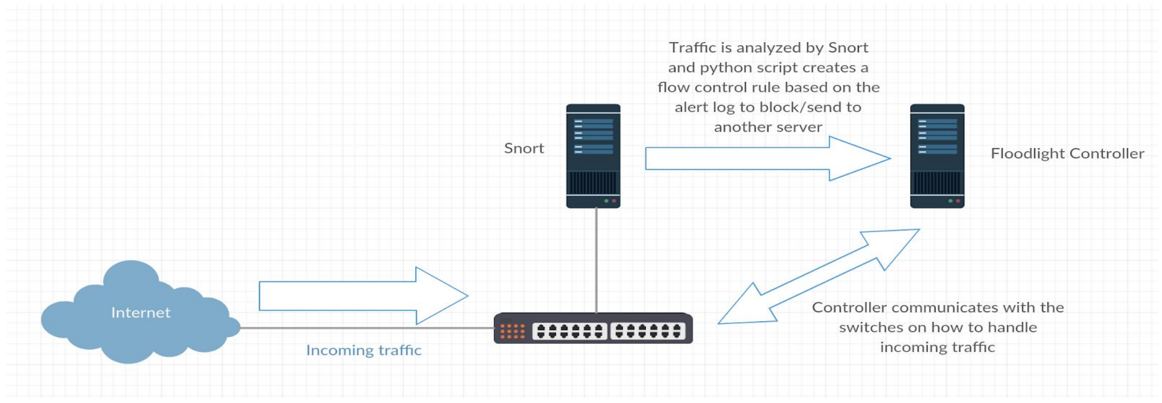
Traffic is analyzed by Snort and python script creates a flow control rule based on the alert log to block/send to another server
Snort
Floodlight Controller
Internet
Incoming traffic
Controller communicates with the switches on how to handle incoming traffic

Figure 1: Visual representation of how Snort and the Floodlight Controller communicate with the network

Once the rules are pushed to the Floodlight Controller, we can see that the overall design made such that once the alert is created that it will have 3 options: 1) block the traffic 2) redirect the traffic to the honeypot server 3) let the traffic continue as normal.

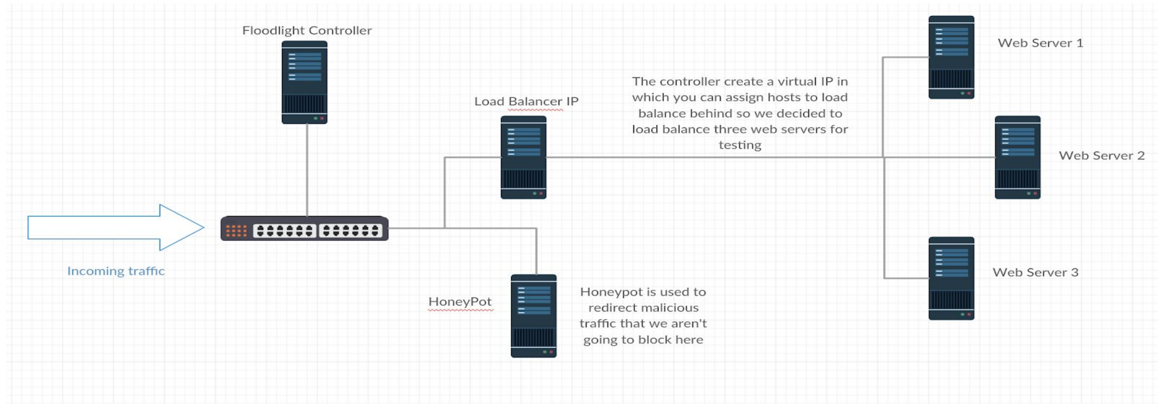
Floodlight Controller
Load Balancer IP
The controller create a virtual IP in which you can assign hosts to load balance behind so we decided to load balance three web servers for testing
Web Server 1
Web Server 2
Web Server 3
Incoming traffic
HoneyPot
Honeypot is used to redirect malicious traffic that we aren't going to block here

Figure 2: Detailed design of the internal network

2.2 Design Analysis

Since the idea of SDN MTD is still somewhat new, we spent a lot of time first semester solely doing research before starting to implement anything. We experimented with a handful of different tools before deciding on Floodlight as our controller. It was the simplest to use and also had the capability for exactly what we wanted. We also tried writing our own custom Floodlight modules so Floodlight itself would recognize the malicious traffic, but that did not work well and did not integrate well which is why we switched to using only Snort as our IDS.

Due to all the experimentation we tried throughout our project, we are confident that our final design is by far the most practical.

3 Specifications and Analysis

3.1 Interface Specifications

Our current test design to mimic a company's production environment will consist of creating a virtual hypervisor, Citrix XenServer, because we noticed that the hypervisor supported the use of Open vSwitches. Through this we will create a backbone of machines within XenServer to test our the virtual switch controller configuration and determine the packet's path with multiple virtual machines running. We will interface the Open vSwitch by using the Floodlight Open SDN Controller to direct traffic to and from our virtual machines.

3.2 Hardware and software

Hardware that we used for this design consisted of a Dell PowerEdge R710 Server. This server is running VMware Hypervisor to allow for creation and configuration of virtual machines. Having our own physical server to work on allows for us to easily create and destroy machines as needed in our design without having to get more hardware.

We use Floodlight Open SDN Controller as our controller because it is straightforward to use and it supports the OpenFlow Protocol which allows us to manage the flow of traffic within our testing network to the machines. We also use Snort as our intrusion detection system to monitor incoming packets and generate alerts based on rules we created. Another tool we use is Kali Linux. We use this for testing to run scans and DDoS attacks against the network.

3.3 Functional Testing

Functional tests will include but are not limited to:

- Analyze packet flow under normal circumstances to compare to our packet flow during tests
- Nmap scans and nikto scans from a Kali Linux box and seeing that packets route to the correct server
- Relieving DDoS pressure by blocking connection from an ip if an overload of packets is sensed
- Attempt to test our system in the Cyber Defense Competition

3.4 Non-Functional Testing

- Availability - The company's provided services will not be hindered by the network design. There should be no outages as well as no unexpected downtime.
- Usability - Customers should see no change in the services they access over the web. For example, if a user is downloading something from the internet and the load balancer switches the hosts, the download should still be finished on the original machine so it does not fail.
- Usability - The system should be simple to set up and add to an existing network.

3.5 Process

Testing of the SDN MTD system consisted of two portions. First, our own testing for a baseline with internal testing of lag due to overhead. Additionally, we attempted to the test results of our system defending from attacks propagated by the *red team* during the Community College CDC that was held in Coover Hall on December 1st, 2018.

To form a baseline to test against for all future applications and to determine the overhead that our new integration and code causes within our system a Cron tab was setup in our testing environment to run metrics over time. The crontab measured response time every 30 minutes over several weeks to analyze packet loads under normal circumstances which provided us with long term, stable numbers on the min, max, and average ping rates. At that point we could test the ping rates while running various tests including those that are redirected to the honeypot such as mapping attacks and those that are simply blocked such as DDoS attacks.

During the Community College CDC held at Iowa State on December 1st our team attempted to set up our software defined network to be tested by the red team. We migrated the Snort machine with the Floodlight controller that our work is based on to the required CDC servers. The stated goal of the competition was to set up administrative servers to be tested against by the red team and to take control of stored simulated files that emulate an Iowa State administrative system, primarily a Canvas box with stored names, birthdates, classes, and other assorted information. With our system in place between the "public" facing connections and the stored information, all malicious traffic should be intercepted and either misdirected or stopped. Ideal results would show that malicious attacks from the red team are unable to reach our machines with generated logs from the Canvas box and alerts from the Snort machine. Unfortunately, due to difficulties with the CDC required machines and their implementation, we were not able to enter this project into the CDC for testing. While we were not able to compete, we were able to set up our SDN MTD system with various servers being protected.

3.6 Results

Through our testing we determined that the packet speed in which a packet transfers to the same server vs load balancer vs a scan is shown in the table above. We can see that when directly pinging the server we get a decent average response time, with a comparable response time using the load balancer. With the scans we determined out of every few hundreds of packets were transferred that we only lost one packet due to when the rule was added and applied to the controller since there was a long enough delay for that packet to be dropped not knowing where to go. The average response times for the packets seemed to have increased during our scanning period to suggest that there is a little more overhead when adding the Floodlight Controller rule in our network for where the destination of where the packet would be going. Below are the times of our results as well as graphs to show the ping times during our testing of our network design:

Packet Test	Min (ms)	Avg (ms)	Max (ms)
Direct	0.34331481	0.66207407	2.70742593
Load Balance	0.34418182	0.68587273	2.42092727
Nikto Scan	0.43866968	0.77452036	5.67220814
Nmap Scan	0.471375	0.719875	1.9785

Figure 3: Table of testing packet delay results

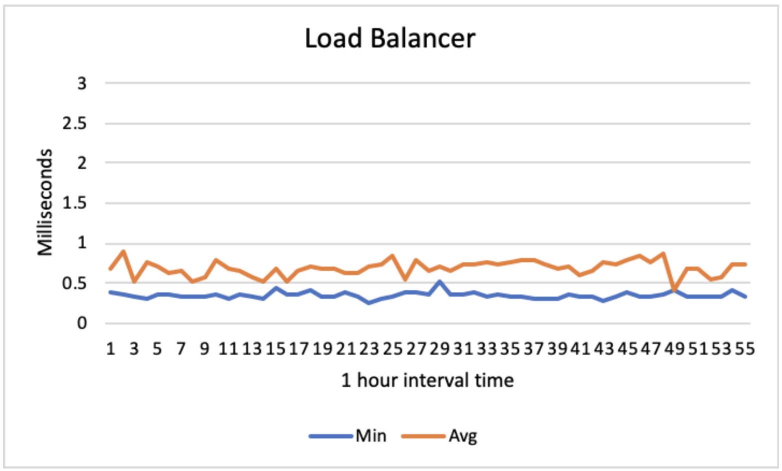

Figure 4: Load Balancer baseline ping graph

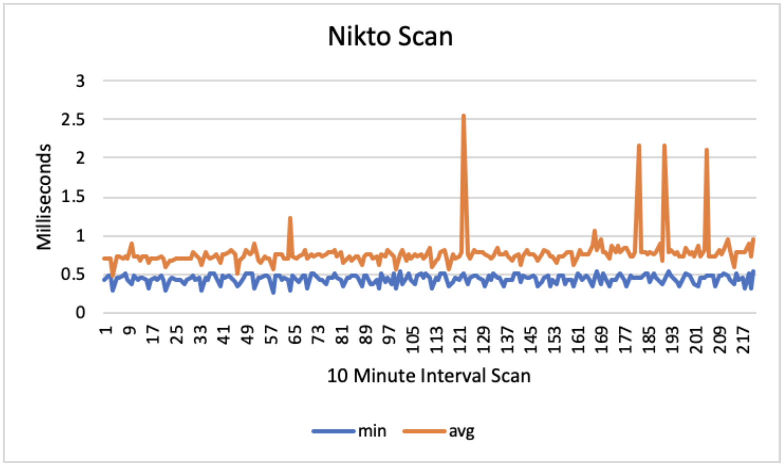

Figure 5: Nikto Scan packet response times

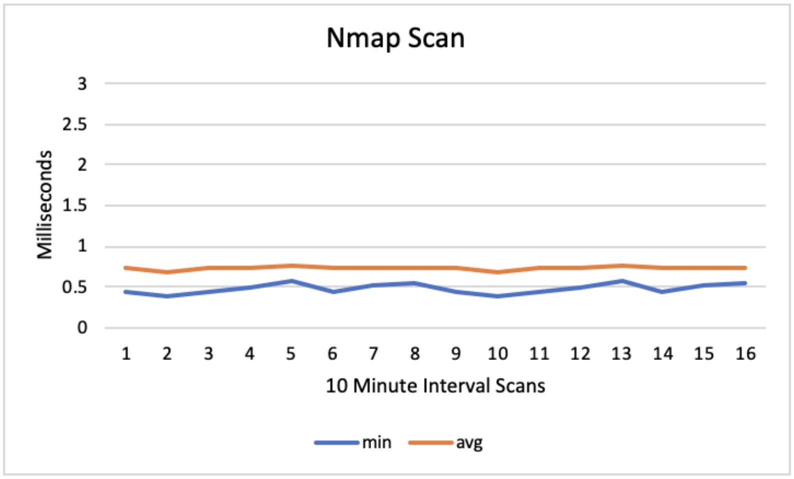

Figure 6: nmap scan packet response times

As mentioned above, our testing with the CDC did not go as planned, but we still got some useful information from it. We were able to set up our system on the competition network without too much difficulty. There are a lot of small components that need to work together in order for this system to work, so it was good to test being able to successfully set it up in a simulated network environment. Not being able to test the functionality, although disappointing, isn't the end of the world. The attacks run by red team are so broad that we suspect the results from our specific testing are more useful than those would have been. Red team will do a variety of attacks that would not necessarily contain just scanning our machines but of other penetration testing tools that our intrusion detection system, Snort, may not have alerted which would result in missed opportunities for the design to be fully tested. This is avoided with our manually testing because we know exactly which types of scans will alert in our intrusion detection system so we can continuously test our network design and make sure that it gets reasonable results from the rules that we created.

4 Closing Material

4.1 Conclusion

With the amount of security risks that static networks can face in today's world, a solution to provide extra layers of security to the network is needed. The goal of our Software Defined Network Moving Target Defense system helps to alleviate this risk. By creating this we are able to monitor, control, and analyze packets that go through a network and minimize the risk of information gathering and manipulate the flow of traffic to protect the network as a whole.The usage of our Software Defined Network Moving Target Defense system will allow for a safer environment for companies to use in their networks while at the same time providing very little overhead so that end users would not notice the small difference in delay for when software defined network rules are created.

4.2 References

[1] Jafarian, J. H., Niakanlahiji, A., Al-Shaer, E., & Duan, Q. (2016). Multi-dimensional Host Identity Anonymization for Defeating Skilled Attackers. Proceedings of the 2016 ACM Workshop on Moving Target Defense - MTD16. doi:10.1145/2995272.2995278

[2] Kampanakis, P., Perros, H., & Beyene, T. (2014). SDN-based solutions for Moving Target Defense network protection. Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014. doi:10.1109/wowmom.2014.6918979

[3] Mahler, D. (2014). Netfool Networking. Retrieved from https://www.youtube.com/user/mahler711

[4] Okhravi, H., Rabe, M. A., Mayberry, T. J., Leonard, W. G., Hobson, T. R., Bigelow, D., & Streilein, W. W. (2013). Survey of Cyber Moving Target Techniques. doi:10.21236/ada591804

[5] Skowyra, R., Bauer, K., Dedhia, V., & Okhravi, H. (2016). Have No PHEAR. Proceedings of the 2016 ACM Workshop on Moving Target Defense - MTD16. doi:10.1145/2995272.2995276

[6] Stakhanova, Natalia; Basu, Samik; and Wong, Johnny S., "A Taxonomy of Intrusion Response Systems" (2006). Computer Science Technical Reports. Paper 210. http://lib.dr.iastate.edu/cs_techreports/210

[7] Zhuang, R., Bardas, A. G., Deloach, S. A., & Ou, X. (2015). A Theory of Cyber Attacks. Proceedings of the Second ACM Workshop on Moving Target Defense - MTD 15. doi:10.1145/2808475.2808478

[8] Zhuang, R., S. A., & Ou, X. (2015). Towards a Theory of Moving Target Defense. Proceedings of the First ACM Workshop on Moving Target Defense - MTD 14. doi:10.1145/2663474.2663479

[9] Citrix Systems, Inc., Documentation. https://xenserver.org/overview-xenserver-open-source-virtualization/documentation.html

4.3 Appendices

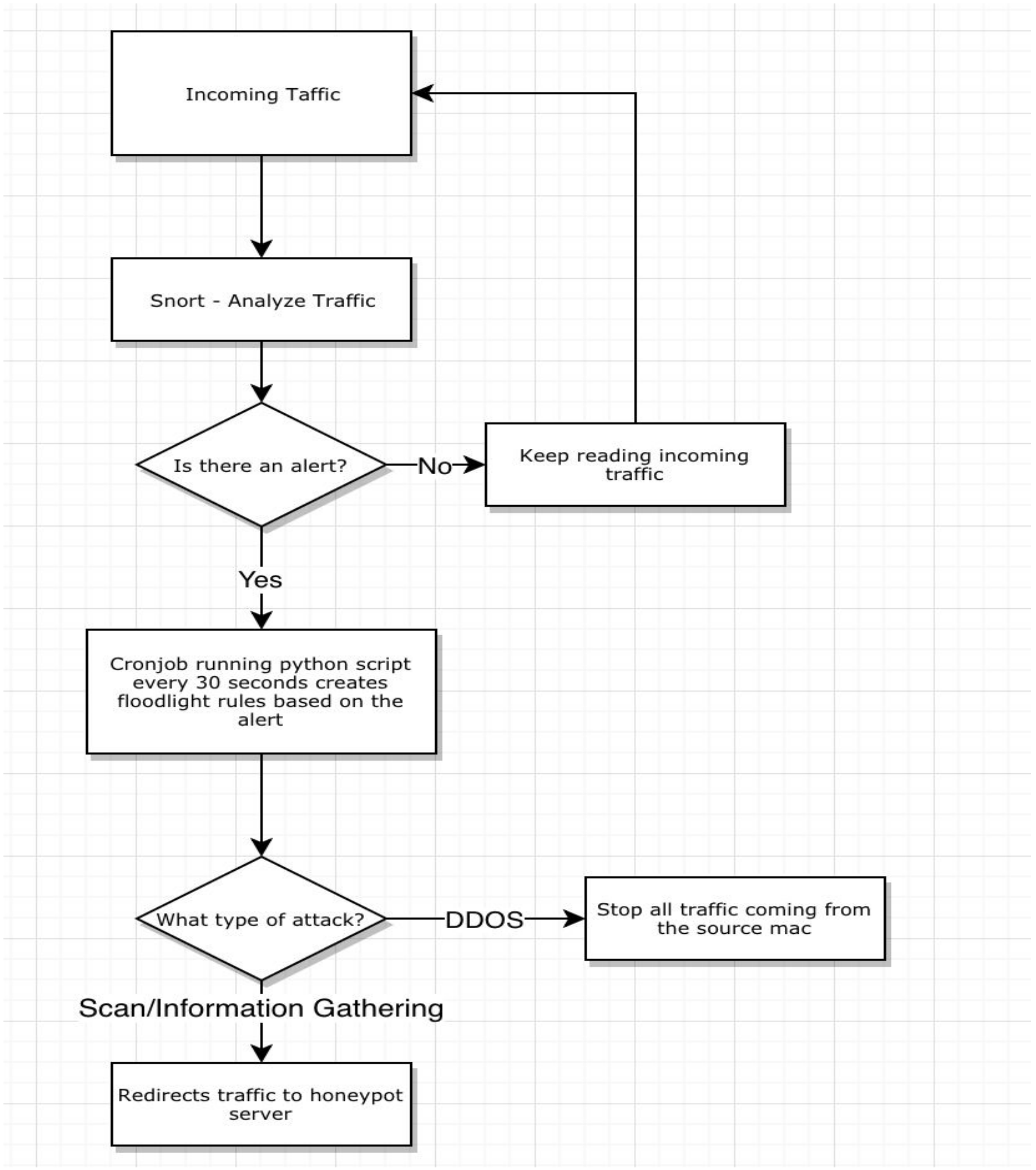
Incoming Taffic
Snort - Analyze Traffic
Is there an alert?
No
Keep reading incoming traffic
Yes
Cronjob running python script every 30 seconds creates floodlight rules based on the alert
What type of attack?
DDOS
Stop all traffic coming from the source mac
Scan/Information Gathering
Redirects traffic to honeypot server

Figure 7: Block diagram of how our implementation is designed based on our servers