ISSN 2355-0082

Naïve Bayes Anomaly Detection System Design On Openflow Network

Nehemia Edbertus[1], Samuel Hutagalung[2], Hargyo Tri Nugroho[3]
Multimedia Nusantara University: Departement of Computer Engineering, Tangerang, Indonesia

Accepted on 11 July 2019
Approved on 20 December 2019

***Abstract*— One of the generally launched attacks is Distributed Denial of Service that renders its target unable to provide its service. Gaussian Naïve Bayes Classifier is one out of several techniques used in detecting those attacks by classifying network traffic in a window as an attack or a normal traffic based on normal distribution previously calculated from normal and attack traffic datasets. This research focuses on mitigating SYN Flood Type DDoS attacks on OpenFlow Network using Zodiac FX as a switch. The developed system utilizes OpenFlow Protocol to apply flow rule in switch's flow table in order to detect and mitigate SYN Flood attacks in real-time. Applied mitigation procedure is to divert incoming packets into SYN Proxy so that only legitimate TCP packets are able to reach the server. The results show that the system has a bandwidth of up to 60Mbps under normal condition and 5,03Mbps under attack. Maximum malicious packets that could reach server before it is diverted to SYN Proxy is estimated to be 400 packets and not affected by the number of attacks, assuming that the flow rule sent by the controller are enacted immediately.**

Index Terms*— *DDoS Mitigation*, *Naïve Baye*, *OpenFlow Protocol*, *SYN Proxy*, *Zodiac FX

I. INTRODUCTION

With the development of services and content available on the internet, network security has become a very important part in it. The advancementin network security is due to increasing number of various crimes in cyberspace. One type of attack that is well known and is still frequently launched is Denial of Service (DoS) which is renders its target unable to run its services [1].

Symantec Corporation in its annual report, 2017 Internet Security Threat Report, published in April 2017 describes a new trend of DDoS attacks that utilize various Internet of Things (IoT) devices that have a low level of security as a tool to launch DDoS attacks. One of the notable incidents was when Mirai (botnet consisting of IoT devices) managed to do DDoS on a DNS company called Dyn, which resulted in obstruction of access to several well-known websites such as Netflix, Twitter and Paypal [2]. Another report on DDoS attack fromKasperskyLab for the fourth quarter of 2017 stated that among all DDoS attack methods, SYN Flood was reported as the most frequently launched attack method with a percentage reaching 55.63% of all existing DDoS attacks as shown in Figure 1 [3].

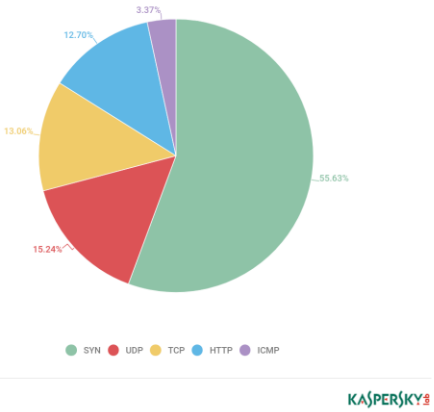

Figure 1.Distribution of DDoS Attacks by Type in Q4 2017 [3]

Various solutions for detecting DDoS attacks, especially the type of SYN Flood attack have been developed and one of them is "Implementation of Count-Min Sketch on Naïve Bayes Anomaly Detection System". The research compares the performance of an anomaly detection system using a count-min sketch data structure with a data structure linked to the Naïve Bayes Classifier to read patterns of captured network traffic and determine the occurrence of attacks. Another related research is [4] which has successfully implemented Bloom Filter to detect DDoS on OpenFlow networks. In addition, [7] has conducted research on HTTP Flood detection using Naïve Bayes and can detect SYN Flood with 99.8% accuracy. Lastly, [6] utilizeOpenFlow controller's ability to mitigate attacks by diverting and enforcing special actions against the suspected maliciouspackets. By manipulating incoming packets to be sent to its destination and detection system simultaneously, each incoming packet can be analyzed and forwarded concurrently.If analyzed packets found to exceed the calculated threshold, it can be concluded that the flow is an attack.This research proposed to do mitigation based on current network condition so that under normal circumstances, all incoming packets can directly access the server, otherwise it will be filtered before reaching the server.

With those goal in mind, this research is built on OpenFlow protocol which in recent years have been gaining more attention and is predicted to be the future of network technology. While still being under continuous development, this protocol is fully functional. Using this protocol, user can manipulate switches behavior by programming flow tables in various switch that supports this protocol [6] and [7] and consistently produce identical behavior.

In order to detect SYN Flooding, this research used gaussian naïve bayes classifier, a machine learning technique to classify a sets of data based on their feature. Before used to classify, it needs to be trained using existing data belonged to specific classes. While assuming each feature is has no corelation or independent from one another, most cases were violating this assumption, but even with corelation or dependencies between its feature, Harry Zhang proves in his research [4] that naïve bayes could still work optimally. In gaussian naïve bayes, it used normal or gaussian distribution in representing its probabilistic value from each feature for each class. First, estimating normal distribution from training data sets by calculating mean(μ) and variance(σ^2) for each feature, then prior probability($p(Cn)$) of each classes.

$$p(C_n) = \frac{N_c}{\Sigma(N_C)} \quad (1)$$

$$p(X_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} exp\left(-\frac{(X_i-\mu_C)^2}{2\sigma_C^2}\right) \quad (2)$$

$$p(X|C) = p(X_1|C) * p(X_2|C) * \ldots * p(X_n|C) \quad (3)$$

$$P(C_n|X) = \frac{p(X|C_n) * p(C_n)}{p(X|C_1) * p(C_1) + p(X|C_2) * p(C_2) + \cdots + p(X|C_n) * p(C_n)} \quad (4)$$

Equation (1) represents prior probability of a class based on its training data sets, where C is class and N is number of event for each class. Equation (2) shows probability of a feature(X_i) for class(C). Equation (3) shows probability of a data(X) for class(C), where based on naïve bayes assumptions of independencies, every feature for class from equation (2) is multiplied. Equation (4) is the final probabilistic value, predicting the percentage of a class(C) for the data(X) where the result from equation (3) is multiplied with its classes prior proability from equation (1) and divided by the sum of every result from equation (3) multiplied with equation (1) for each class. The class probability with highest value is the predicted class for the data.

$$Multiplier = \frac{2}{TimePeriods + 1} \quad (5)$$

$$EMA = (CurrentValue - PreviousEMA) * Multiplier + PreviousEMA \quad (6)$$

While to produce a false-positive-proofed system, exponential moving average could be used to determine current trends while keeping track of several previous record of current condition, but put extra weigh on current condition as seen in equation (5) and (6).

II. RESEARCH METHOD

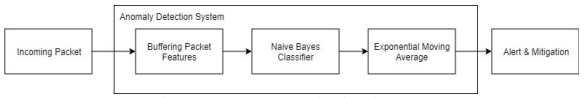

Figure 2. System Block Diagram

Figure 2 shows system's block diagram. Incoming packet will be copied and forwarded to anomaly detection system to be analyzed by extracting its features in buffer. Buffered packet features will then sent to naïve bayes classifier to be classified as normal or malicious flow. If the number of malicious flow is greater than normal flow, the buffer would be concluded as an attack attempt and vice versa. This current condition would then further calculated in exponential moving average module so that the final outcome of current condition would have less false positive. This final outcome will determine which rule enacted by the OpenFlow switch.

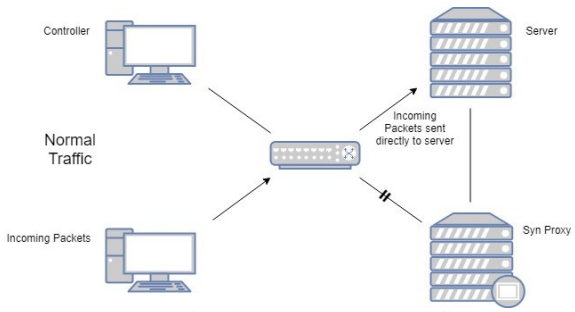

Figure 3. Packet Flow Under Normal Condition

Figure 3 shows incoming packetswill be forwarded directly to server by OpenFlow Switch and there is no means to communicate with SYN Proxy directly from the switchunder normal condition.

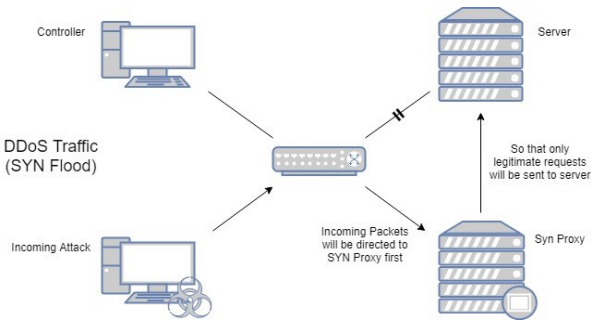

Figure 4. Packet Flow Under Attacked Condition

However, when under attack, a new set of rules will be enforced, preventing all incoming packet from directly communicating with server by diverting those packets to a SYN Proxy server as shown in figure 4. SYN Proxy server would be filtering all incoming packets and forward legitimate request only so that legitimate user can still access the server and the server wouldn't receive devastating number of

malicious packets that potentially exhaust its resources and crashes the server, resulting in greater loss.

```
Main()
    set UNDER_ATTACK = false
    set BUFFER_TRESHOLD = 100
    set MOVING_AVERAGE_VALUE = 0
    set MOVING_AVERAGE_TIME_PERIOD = 10
    set MOVING_AVERAGE_MULTIPLIER = 2/(MOVING_AVERAGE_TIME_PERIOD+1)
    switch.start()
    controller.start()

switch.onPacketReceived(incomingPacket)
    forwardBasedOnRule(incomingPacket)

procedure forwardBasedOnRule(packet)
    if(packet is tcpPacket)
        cloneToController(packet)
    if(UNDER_ATTACK)
        forwardToSYNProxy(packet)
    else
        forwardToServer(packet)
```

Figure 5.Handling Incoming Packet Pseudocode

This research utilize OpenFlow Switch capabilities to forward packets based on enforced flow rules. Figure 5 shows how system is initialized and handles incoming packet events on switch. If the packet is a tcp packet, it will be cloned and forwarded to controller while simultaneously forwarded it to its destination based on system UNDER_ATTACK state (to server if UNDER_ATTACK set to false and to SYN proxy if true).

```
procedure BufferTCPPacket(packet)
    set flow (packet.srcIP, packet.destIP, packet.dstPort)
    buffer[flow].append(packet.features)
    increment bufferSize

controller.on(packetReceived)
foreach packet receivedBy controller
    BufferTCPPacket(packet)
    if(bufferSize == BUFFER_TRESHOLD)
        AnalyzeBufferWindow(buffer)
        reset buffer
```

Figure 6. Buffering Packet Pseudocode

Figure 6 explains how cloned TCP packets is buffered in controller. Each received packet's feature will be buffered and when buffer size reach predetermined threshold (BUFFER_TRESHOLD in Figure 5), buffer will be analyzed and then resetted.

```
procedure AnalyzeBufferWindow(buffer)
    set normalFlow 0
    set anomalyFlow 0
    foreach flow in buffer
        set class = GaussianNaiveBayes(flow.features)
        if(class == anomaly)
            increment anomalyFlow
        else
            increment normalFlow
    if(anomalyFlow > normalFlow)
        condition = anomaly
    else
        condition = normal
    set UNDER_ATTACK = ExponentialMovingAverage(condition)
```

Figure 7. Analyze Buffer Pseudocode

Figure 7 shows how buffer is analyzed by classifying each flow in buffer using Gaussian Naïve Bayes Classifier. Condition will be set to anomaly if anomalyFlow counts more than normalFlow, and vice versa. System UNDER_ATTACK state will be determined by ExponentialMovingAverage calculation based on current condition as shown in figure 8 below.

```
procedure ExponentialMovingAverage(condition)
    if(condition == anomaly)
        set currentValue = 1
    else
        set currentValue = 0
    set MOVING_AVERAGE_VALUE = (currentValue-MOVING_AVERAGE_VALUE)*MOVING_AVERAGE_MULTIPLIER + MOVING_AVERAGE_VALUE
    if(MOVING_AVERAGE_VALUE>0.5)
        return true
    else
        return false
```

Figure 8.Exponential Moving Average Pseudocode

Figure 8 shows how system determined UNDER_ATTACK state by calculating current MOVING_AVERAGE_VALUE using exponential moving average. If current MOVING_AVERAGE_VALUE is more than 0.5, UNDER_ATTACK will be set to true and if less than 0.5 to false. By setting MOVING_AVERAGE_TIME_PERIOD to 10 event and BUFFER_TRESHOLD to 100 packets, maximum number of anomaly packets received by server can hypothetically be calculated as shown in Table 1 below.

Table 1. Exponential Moving Average Calculation

Event #	anomaly_value	EMA	condition	anomaly_packet	anomaly packet received by server	anomaly packet diverted
1	False	0	Normal	0	0	0
2	True	0.181818	Normal	100	100	0
3	True	0.330579	Normal	100	100	0
4	True	0.452292	Normal	100	100	0
5	True	0.551875	Anomaly	100	100	0
6	True	0.633352	Anomaly	100	0	100
7	True	0.700015	Anomaly	100	0	100
8	True	0.754558	Anomaly	100	0	100
9	True	0.799184	Anomaly	100	0	100
10	True	0.835696	Anomaly	100	0	100
11	True	0.865569	Anomaly	100	0	100

Assuming incoming attacks is a bursting 1000 TCP SYN packets with the speed of 10 packets per second and all flow rules sent by controller are enacted immediately, then maximum number of anomaly packets received by server can be estimated to 400 packets or 4 anomaly events before flow rule to divert incoming packets to syn proxy enactedas shown intable 1. Event #2 up to #11 is an attack, on event#5, EMA value pass over 0.5 and acknowledged as an anomaly. This change on condition triggers controller to enact a mitigation rule on OpenFlow switch and from event #6 onward, all packets is diverted to SYN Proxy until EMA value return under 0.5 and triggers controller to revoke mitigation rule on OpenFlow switch.

This research use HPING3 to generate SYN Flood attack to be used in training naïve bayes classifier as anomaly traffic dataset and testing. Testing will be conducted real-time by client accessing a http server under normal condition and under attack by HPING3 SYN Flood condition. System throughput will be tested using iperf on normal and mitigation rule. Testing will be conducted using a laptop with specification Intel core i7 processor, 8GB RAM running Ubuntu Linux VMWare on top Windows10 as OpenFlow controller, 2 PC with specification Intel core i5, 8GB RAM running Windows10 as client and server and another PC with same specification as client/server running Ubuntu Linux VMWare on top of Windows10 as SYN Proxy, and a Zodiac FX as OpenFlow switch.

III. RESULTS AND ANALYSIS

A. *Mitigation*

Using HPING to generate SYN Flood attacks with random source IP designated to the server, the results is shown in figure 9 and 10 below.

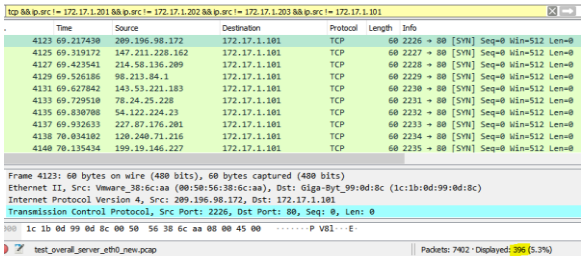

Figure 9. Captured SYN Flood Received by Server

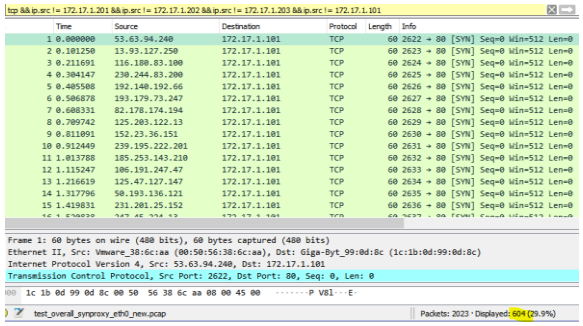

Figure 10. Captured SYN Flood Diverted to SYN Proxy

Out of 1000 SYN flood packets sent, 394 packets received by server and the other 604 packets was successfully diverted to SYN Proxy. This results may vary depending on initial configuration on BUFFER_TRESHOLD and MOVING_AVERAGE_TIME_PERIOD as calculation in table 1 will change accordingly.

B. *Equations*

Using iperf, system's throughput is estimated under normal rule and mitigation rule as shown on table 2 below.

Table 2. System Throughput Test Result

Condition	Speed (Mbps)
Normal	60
Under Attack	5.03

This results shows the system have a throughput of 60Mbps on normal condition and the results mainly caused by Zodiac FX specification which has a 10/100Mbps ethernet port. Another factor that affects this result is the complexity of the flow tables at the time as more complex rules enacted, the slower transfer rate on OpenFlow switches gets. When mitigating attacks, the system has a throughput of 5.03Mbps. This gap between normal and under attack system performance is affected by Network Address Translation applied by SYN proxy to connect client and server when under attack. This results may vary when using SYN proxy, controller or OpenFlow switches with different specifications.

IV. CONCLUSION

From this research developed system can detect and mitigate SYN Flood attacks on OpenFlow networks in real-time. The developed system has a throughput of 60Mbps under normal conditions and 5.03 Mbps when under attack. This speed can vary by using switches or controllers with different specifications. And maximum number of SYN Flood packets that can reach the server is estimated to be 400 packets without being affected by the number of attack packets sent with the condition that all flow rules sent by the controller are immediately applied and the rest will be transferred to SYN Proxy. This number can be tweaked to accommodate user preferences by changing buffer size treshold and moving average time period. Further development can be carried out to detect and mitigate other attacks on OpenFlow networks and use other classification method.

References

[1] C. Patrikakis, M. Masikos and O. Zouraraki, "Distributed Denial of Service Attacks," The Internet Protocol Journal, vol. 7, no. 4, pp. 13-35, 2004.

[2] Symantec Corporation, "Internet Security Threat Report 2017," p. 66, April 2017.

[3] A. Khalimonenko, O. Kupreev and K. Ilganaev, "DDoS attacks in Q4 2017 - Securelist," 6 February 2018. [Online]. Available: https://securelist.com/ddos-attacks-in-q4-2017/83729/. [Accessed 10 March 2018].

[4] P. Xiao, Z. Li, H. Qi, W. Qu and H. Yu, "An Efficient DDoS Detection with Bloom Filter in SDN," in IEEE TrustCom/BigDataSE/ISPA, Tianjin, 2016.

[5] V. Katkar, A. Zinjade, S. Dalvi, T. Bafna and R. Mahajan, "Detection of DoS/DDoS attack against HTTP," in International Conference on Computing Communication Control and Automation, Pune, 2015.

[6] Y.-K. Lai, C.-C. Lee, B.-H. Huang, T. Wellem, N.-C. Wang, T.-Y. Chou and H. T. Nugroho, "Real-time detection of changes in network with OpenFlow based on NetFPGA implementation," Microprocessors and Microsystems, vol. 38, no. 5, pp. 431-442, 2014.

[7] Samuel, S., & Samudera, C, "Rancang Bangun Mekanisme Load Sharing Pada Link Aggregation Menggunakan Software Defined Networking," Ultima Computing : Jurnal Sistem Komputer, 9(1), 41-47, 2017.