VILNIUS TECH
Vilnius Gediminas Technical University

Mokslas – Lietuvos ateitis / Science – Future of Lithuania
ISSN 2029-2341 / eISSN 2029-2252
2023 Volume 15, Article ID: mla.2023.18847, 1–7
https://doi.org/10.3846/mla.2023.18847

Electronics and electrical engineering
Elektronika ir elektros inžinerija

ANALYSIS OF TCP FLOOD ATTACK USING NETFLOW

Vsevolod KAPUSTIN *, Nerijus PAULAUSKAS

Vilnius Gediminas Technical University, Vilnius, Lithuania

Received 26 October 2022; accepted 07 April 2023

Abstract. Traffic analysis is a common question for most of the production systems in various segments of computer networks. Attacks, configuration mistakes, and other factors can cause network increased accessibility and as a result danger for data privacy. Analyzing network flow and their single packets can be helpful for anomalies detection. Well-known network equipment has predeveloped network flow monitoring software. "NetFlow" data collector software "Nfsen" is an open-source way to collect information from agents. Also "Nfsen" is designed for data sorting and dataset for instruction detection system preparation. Prepared data can be split into fragments for artificial intelligent learning and testing. As AI unit can be used multilayer perceptron developed in a python programming language. This paper focused on real-world traffic dataset collection and multilayer perceptron deployment for TCP flood traffic detection.

Keywords: NetFlow, tcpdump, TCP, packet, firewall, traffic, GRE, attack.

Introduction

Nowadays, network traffic analysis is an important way to diagnose network vulnerabilities. Standard intrusion detection systems made by network equipment developers such as Sophos (Sophos Labs), Fortinet (Fortiguard Labs), and others, store intelligent system parameters in private databases that are accessible only for prepaid users of products. Some open-source IPS providers like "Snort" store well-known anomalies detection preprocessing algorithm structures in public repositories (Szmit et al., 2007).

The main goal of integrated systems is to prevent local networks with some production public resources of Denial of services (DoS) attacks, which can cause resources inaccessibility or data processing mechanisms misinterpretation. All these factors can course financial losses and fall in prestige on the business market.

Globally attacks can be divided into two types: passive – data collection and preparation and active – data and systems manipulation by generating anomalies interfering normal operation of systems (Samtani et al., 2019).

This paper will provide an experiment of analyzing popular DoS attack types on the transport layer (relatively to "Open systems interconnect" OSI standards). TCP is used for common application layer protocol encapsulation like HTTP (s) (WEB surfing), FTP (File transferring), DNS (Internet name resolution), and more others. Therefore, TCP is the easiest way to influence the production systems of organizations.

Maybe network security question is common nowadays, but most organizations cannot buy dedicated security systems or deploy open-source filters for private servers.

All network anomalies detection systems are based on a global information database. Data Science algorithms are detecting malware and classify it by updating neuron network parameters (Sophos, 2019). Synchronizing database with network unified threat management devices builds up-to-date algorithms in local internet traffic gateways, that protects private information from attackers.

1. About TCP connectivity

TCP Protocol is a connection-oriented protocol located in the transport layer (the fourth one based on the OSI model). It guarantees data delivery from server to client, by using mechanisms of synchronizations.

TCP header format on transport layer is quite different from other transport protocol UDP. The transport layer contains (Duke et al., 2006): the sequence for opening the TCP connection is called "Three-way handshake" and its

*Corresponding author. E-mail: *vsevolod.kapustin@stud.vilniustech.lt*

goal is to initiate a data transfer "channel" between server and client. For analyzing TCP handshake establishment, can be used network analysis tool "Wireshark".

The connection can be initiated using WEB server service connectivity. HTTP protocol (Application layer) is encapsulated to TCP, so a simple request can initiate the connection. By using "Wireshark" filters handshake can be obtained (Figure 1): Code bit SYN set indicates the first packet of handshakes. Code field of header will be equal to 0x02 (filter value: "tcp.flags == 0x02"). Also, the first handshake packet can be obtained by sequence and acknowledgment number set to zero.

Second is SYNACK packet from server side, that means source address will be set to server address, SYN flag is set and ACK flag is set. Code field will be equal 0x12. Also, relative sequence number is 0 and acknowledgment number is 1. "Wireshark" filter segment looks like "tcp.flags == 0x12 && tcp.seq == 0 && tcp.ack == 1"

Third packet, acknowledgment from client side will be set ACK and sequence and acknowledgment numbers will be set to 1. Filter contains string "tcp.flags == 0x08 && tcp.seq == 1 && tcp.ack == 1".

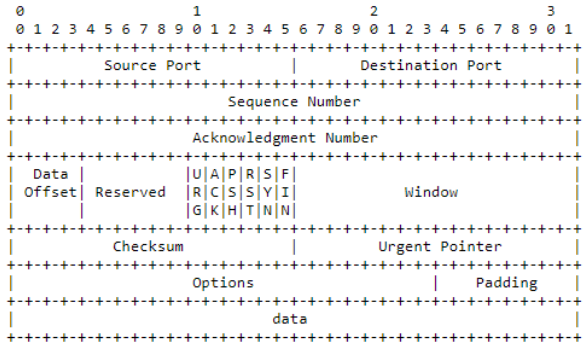

Figure 1. TCP header format

Figure 2 represents the TCP connectivity sequence generated in "Wireshark" software

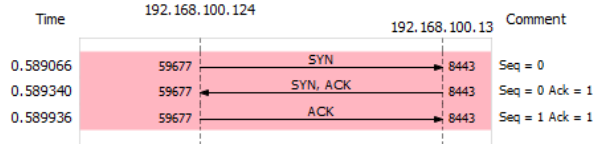

Figure 2. TCP three-way handshake sequential diagram

In this paper, analyzing TCP flood attacks important step is timing analysis. As presented in Figure 2. The time delta between SYN and SYNACK packet is 274 us and 597 us to ACK from the client (test made when the client is connecting to the server via not IPsec based Generic routing encapsulation tunnel and server has 8 hops distance).

2. TCP SYN flood attack with IP spoofing

The TCP flood attack principle is based on sending many packets with spoofed and forged source IP addresses. The server will receive TCP SYN packets and send SYNACK back to not client (attacker) side IP address, and then look for acknowledgment from the client. Since the destination IP of SYNACK is not a connection-initiated device if there will be an active device on this address it will receive the packet and drop it.

While the server doesn't receive the ACK packet, it will send a retransmission of SYNACK and increase the waiting queue as waiting time. During the attack, the queue will increase and inevitably reach the maximum of performance, then will drop connections on a port, that cause Denial of service (DoS). Figure 3 (Jin & Lin, 2011) represents a mechanism of the TCP flood attack, and Figure 4 represents one-time session communication during the experiment described in the "Dataset collection" section.

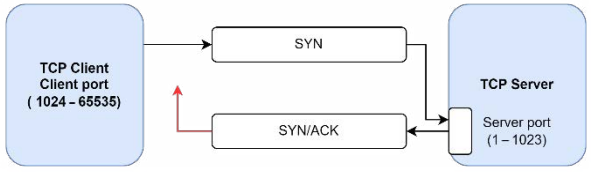

Figure 3. TCP flood attack mechanism diagram

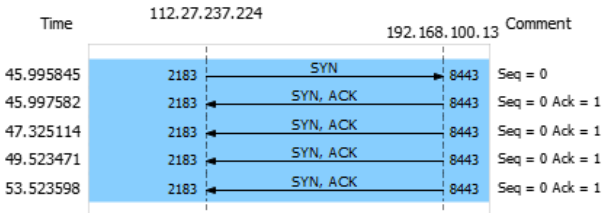

Figure 4. TCP flood attack one session sequential diagram

"Because of the variability of the networks that compose an internetwork system and the wide range of uses of TCP connections the retransmission timeout must be dynamically determined" (Raja & Vasudevan, 2017). As shown in Figure 4 experimental victim system has three retransmission packets send, and the queue elapsed time is about 8.5 s. In flood, the attack can be sent about 400 SYN packets per second (sec. "Dataset collection").

So, TCP flood can be described as SYN packets with a random source port and IP address sent to victims one after another. The time between packets depends on attackers' system performance, data transfer line bandwidth, and node's conntrack capabilities.

3. Dataset collecting environment

Dataset collecting environment is based on 5 active devices (systems): TCP flood flow generating system based on Kali GNU/Linux, victim machine with WEB server preinstalled, CentOS Linux 7 machine with NetFlow collector and organizer installed, Fortigate 80C physical appliance with turned on "NetFlow v5" sampler on "internal" interface and administrator machine with Windows 11 operating system installed (Figure 5). So, let's describe all system roles.

Nfsen collector. The Nfsen collector is a virtual machine located in Datacenter dedicated server with installed Nfsen software. Data collection principle (Figure 6) based on listening packets on UDP port 9995. When a UDP packet is received, a process called Nfcapd checks the packet source IP address and stores NetFlow data in the file. Database hierarchy is collection of directories (structure: /year/month/day) and files that collect 5 minutes data (filename: nfcapd<year><month><day><hour><minute rounded by 5>). For this work there is no script for continuous data processing, all files are generated by the administrator.

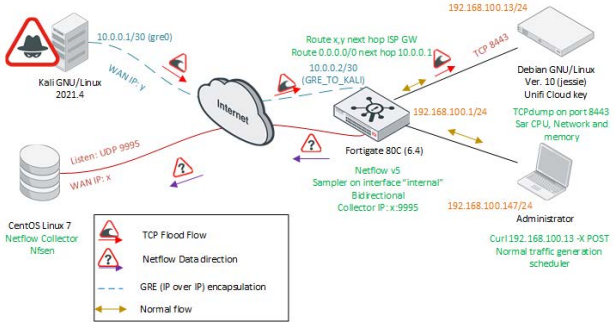

Figure 5. Diagram of dataset collecting environment

Command "nfdump -w /tmp/<Dataset file name>.txt port 8443" (collecting all flows to and from port 8443 and storing to file located on /tmp/file.txt) generates a raw data dump file for subsequent processing. Python language is used for collecting and processing data. Script workflow is following execute "nfdump" command using "sys" package, then read the raw file and convert it to Python dictionary and "pandas" data frame to filter data and convert raw time and date to Linux timestamp (Timestamp displays a UNIX time as well as the current time and date in both coordinated universal time (Zeigermann, 2016)).

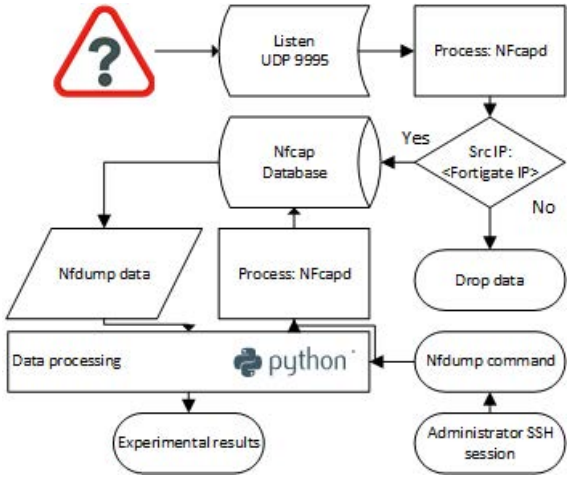

Figure 6. NetFlow processing flow diagram

TCP flood generator. Generating machine is also a virtual machine with installed "Hping3" software. "Hping3" is a network tool able to send custom TCP/IP packets and to display target replies like the ping program does with ICMP replies. Flood is executed using an "hping3" command with a flood argument and destination IP address specification.

Fortigate 80C Appliance. The appliance is a physical device located in the test environment. In this work are usable two physical and one subinterface. Physical interfaces: "internal" – attached to LAN segment of the network; "wan1" – an external interface that internet connection established to. On an external interface configured IP over IP tunnel (GRE) with Kali Linux machine for hiding the IP spoofing from global ISP filters as storing spoofed IP to the second IP header that is encapsulated to the original. On the "internal" interface configured NetFlow sampler in both directions, that means all flows from and to LAN network will be collected and sent to NetFlow collector.

Victim WEB-based controller. IT is a physical device directly connected to the Fortigate appliance. On TCP port 8443 listen to the "Nginx" WEB server with access to the "Ubiquity Unifi" controller WEB interface. During the attack on the machine will run the "tcpdump" packet capturing service with collection to "pcap" file for downloading and reading with "Wireshark" and "Sar" (Sysstat) system statistics capturing (collecting) service. During the work was tried to use monitoring software "Zabbix", but the sampling interval is too large for collecting necessary CPU utilization values.

4. NetFlow data preprocessing

"Nfdump" application gives raw data of collected flow that is impossible to use in algorithms or manipulate using "Pandas" and other predefined Python programming language modules. Also, timestamps are defined in a universal format, that is not used for data processing therefore why all timestamps are converted to UNIX format. In sequence, all data rows are filtered on unusual information such as Xstream parameters.

Python script sequentially checks each row of data and stores formatted rows to the dictionary with predefined headers. All data preprocessing steps is displayed in the Figure 7 diagram.

The second step of preprocessing is to analyze all the flows and insert flag variables to each row. The flag will be a signal for Multilayer perceptron and the stream will be classified as a TCP flood attack.

The algorithm base is to calculate time intervals between packets, that initiates TCP connection (First packets from same source IP).

TCP SYN flood is not possible to obtain in NetFlow data of many devices because there are no presented TCP flags. Parameters presented in NetFlow are timestamps, source, and destination IP addresses and ports.

The main goal of NetFlow data analysis is to dynamically associate durations of flows, packet number per flow, and several flows. TCP flood characteristically has a very short duration of flows (experimentally measured less than

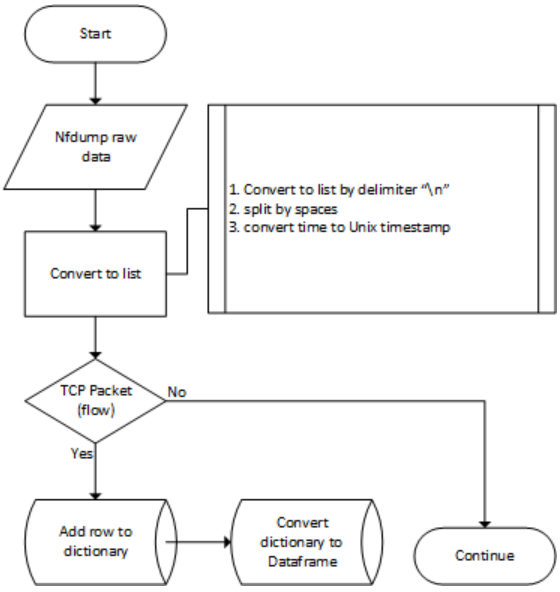

Figure 7. NetFlow data preprocessing flow diagram

50 msec) and a small number of packets. Packets number per flow depends on server TCP retransmission count. For this work experiment, the server retransmits packets 5.

NetFlow data fields are presented in Table 1 (FreeBSD Community, 2005).

Table 1. NetFlow dataset structure

Field	Description
Time first seen	The time that first packet of flow was captured by NetFlow sampler
Duration	Time delta from first packet transition start to last packet transition end
Source IP address and port	Source IP address and port number
Destination IP address and port	Destination IP address and port number
Packets	The count of packets per flow
Bytes	All transferred data per flow in bytes
pps	Packet per second value
bps	Bits per second value
Bpp	Bytes per packet value (headers included)

The data preprocessing task is inserting a new column to the dataset, that represents the TCP flood flow flag. This is necessary for the MLP classifier training phase. In the split-train-test algorithm (Dobbin & Simon, 2011) this value will be used for the training phase. In the test phase, the flood flag will be eliminated.

5. Dataset creation

The collection of the dataset is based on combining malicious traffic and base traffic together. For traffic generation used "bash" script (Table 2). Scripts start the "hping3" flood process and store the process identification to a variable. While the attack is producing, the attacker machine generates base traffic to access HTTPS content.

Table 2. Traffic generation script

```
#!/bin/bash
# Attack generation script
# Create hping3 TCP SYN packet sending process, a process in the background
hping3 -S -i 1u -p 8443 10.100.100.2 --rand-source &
# Storing process ID of hping3 flood attack in variable ps
ps="$!"
# Declare counter variable. Counter sets loop iteration count
i=50
# Loop that executes standard HTTP GET requests to victim (Base traffic generation)
while [ $i -ne 0 ];do
        ## Imitate successful handshake traffic
        curl -k -X GET https://10.100.100.2
        ## Decrement counter by one
        i=$(( i-1 ))
        sleep 0.1
done
# Kill hping3 process after execution
echo $ps
kill $ps
```

After scripts initiation traffic has been collected by active "netflow" collector service that is in default router of victim. All traffic is sent to NetFlow collector by "Netflow v9" protocol. Then using command "nfdump -R nfcapd<start_time>:nfcapd<end_time> -o extended >> dataset" raw dataset data has been stored in file for future processing by Python application.

After NetFlow data preprocessing all data is stored to "Pandas" dataframe. The dataframe is given to Adaptive threshold algorithm (Bradley & Roth, 2007) to append dataframe with parameters.

6. TCP flood attack detection algorithm

The TCP flood attack detection is based on Adaptive threshold algorithm while processing a flows duration and data count per packet, measured in Bytes. Third check is based on source IP connections flags check. The adaptive threshold formula is provided in Equation (1):

$$I(x,y) = f(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1). \quad (1)$$

The algorithm adjusts the exponential weight, which indicates exponential average of flow duration and bytes count per flow. If flow duration is shorter that exponential weight value, algorithm returns anomaly by flow duration flag. Same detection is provided by packet data count.

Second algorithm is based on TCP flags check. Dataset records are grouping by source IP address, destination IP address and destination port. If one group don't have packets flagged by FIN TCP flag, that means TCP session was not processed and closed properly so that means a potential attack. Algorithm diagram provided in Figure 8.

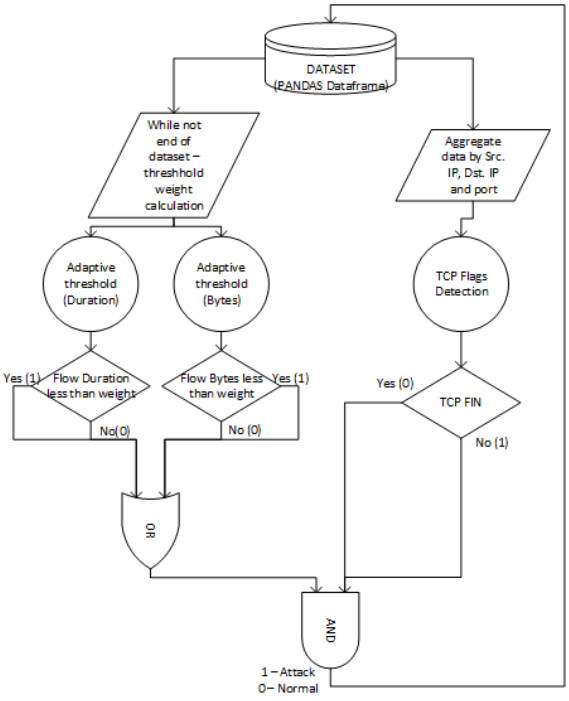

Figure 8. NetFlow data classification algorithm

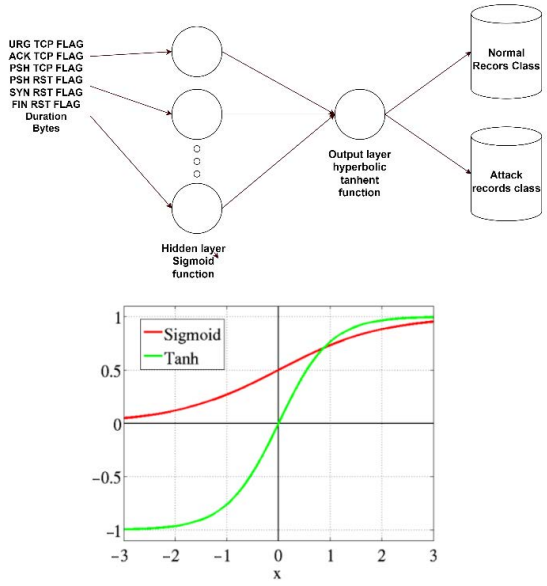

Figure 9. NetFlow classification neural network model

All dataset records are classified as attack and normal traffic. Classification based on algorithms combined by AND function. Hence for attack detect, adaptive threshold and TCP flag detection algorithms should provide attack set signal. After algorithms classifies the traffic, dataset is replenished by detection value.

7. Split-train-test algorithm using MLP for traffic classification

The split-test is a technique for evaluating the performance of machine learning algorithm.

Firstly, dataset has being splinted into two parts for evaluating. First part used by Multilayer perceptron training, and a second part for testing classification efficient. In that work used Neural network model with multilayer perceptron, with 8 inputs (TCP Flags, Duration, Bytes), 100 neurons in hidden layer and one neuron in output layer (Figure 9). Output layer indicates classification results: if one provided by output – attack is detected, if zero traffic are classified as normal. Other flow parameters have been directly connected to the output.

Neural network train phase is based on back propagation algorithm, that updates synaptic weights using two ways check. Hidden layer consists of 100 neurons, each of takes 8 inputs from dataset.

Hidden layer activation function is Sigmoid, and output layer activation functions in hyperbolic tangent.

8. Results and discussions

Dataset created by collecting information of flows from network forwarding equipment is the input of classification algorithms. Collected samples represents real-time Traffic of home user work with some artificial TCP SYN flood traffic generation using Linux machine with installed TCP penetration network tool "hping3".

Dataset contains 1477959 NetFlow samples. Part of dataset contains normal traffic (WEB browsing, file transfer etc.) records. Other part of dataset contains injected by special partial flood packets. Considering to fact, that flood generator is remote Linux Machine TCP SYN flood must be generated partially with 30s discretization and duration not longer than 5s, because internet service provider equipment detects flood and block the server on link layer by blocking traffic to machine.

8.1. Adaptive threshold algorithm

Base collected dataset is not ready for perceptron training. In that case dataset should be additional analyzed by TCP SYN flood detection algorithm. The algorithm based on exponential weight of parameters calculating (Eq. (1)).

Parameters using for classification is TCP header flags, flow duration and bytes per second transfer trough the network.

Flow samples are comparing with exponential weight. If all parameters are less than exponential weight, packet (flow) can be interpreted as TCP SYN flood attack.

8.2. Split-test-train algorithm

After preparation of dataset, split-test-train algorithm using for multilayer perceptron training and trained MLP testing for efficiency measurement.

During the experiment, MLP test and train parts of dataset was partially splitting for different percentage of dataset samples. Results can provide information of classification network dynamic efficient parameter.

Multilayer perceptron network performance is provided in Table 3.

Table 3. Experimental results of NetFlow dataset using MLP

Training samples	Test samples	Classification efficiency
		Multilayer perceptron
1182367	295591	95.3%
886775	291184	91.5%
738979	738980	88%
295592	1182367	85.6%

8.3. Victim machine resources utilization during attack

During experiment, also NetFlow samples are represented to graph using Nfsen (Figure 10). Victim machine was recorded CPU utilization during attack. As provided in graph (Figure 11) CPU utilization is growing, but don't reach 100% in addiction of throughput of intermediate network equipment. Narrow data transition channel causes restriction of complete DoS TCP SYN flood attack. As seen in graph (Figure 12) growing up TCP queue to service causes I/O errors, so no more connections can be handled. This cause temporary traffic decrease (Figure 12).

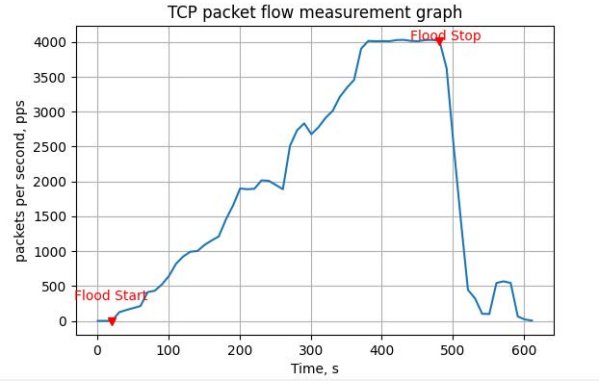

Figure 10. Packet per second graph recorded during experiment

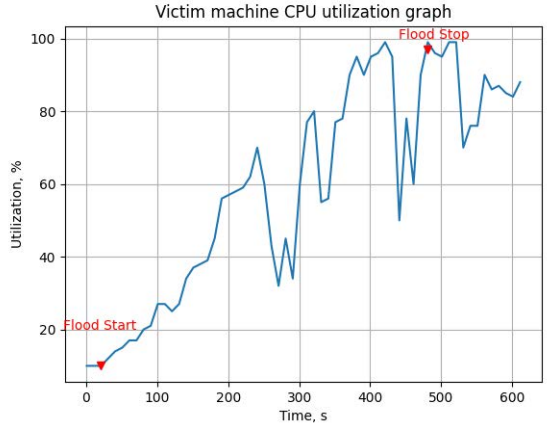

Figure 11. Victim CPU utilization graph

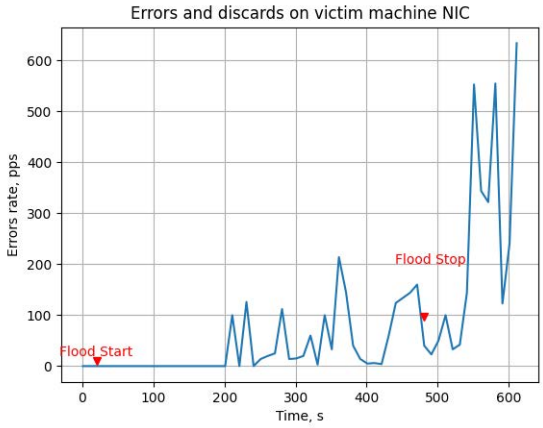

Figure 12. Victim errors and discards graph on network interface

Finally, interpreting the results can be found correlation between CPU utilization and packet processing, also experimental machine can provide capabilities for 4000 packets per second processing. As a result, DoS successfully implemented after 500 s. of flood traffic.

8.4. Comparison of the proposed model

Maximum performance is reached on split-test-train model when used 1182376 (80%) samples for training and 295591 (20%) samples for testing. Sahi and others (2017) work explains more artificial intelligence for TCP SYN flood DoS attack detection and prevention. Comparison of models are presented in Table 4.

Table 4. Artificial intelligence networks efficiency comparison

Model	Efficiency, %
Work model (split-test-train algorithm) multilayer perceptron	95.3
Lease squares support vector machines model	96
Naïve Bayes	96
K-nearest	85

Comparison information shows that efficiency of used model is not the best for such problem classification, but the percent delta shows that the gap is not so big.

Conclusions

DoS attacks is most frequent nowadays. Development of tools for transport protocol attack detection attracts a lot of attention from cyber security specialists. One of the problem-solving methods is traffic collection by NetFlow protocol and processing using algorithms. NetFlow sample dataset provides basic network and transport layers protocols information. Attacks can be obtained by processing such as network layer information like IP addresses and other L3 traffic protocols.

NetFlow data can be preprocessed by dynamic algorithms like dynamic threshold algorithms during is calculated exponential weight. Using exponential weight by comparing sample values, traffic can be classified to attack and normal types.

For classification purposes multi layer perceptron can be used. Using static dataset, neural network can adjust synaptic weights during learning, and classify network traffic types, without using computing resources matching algorithms.

References

Bradley, D., & Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of Graphics Tools*, *12*(2), 13–21. https://doi.org/10.1080/2151237X.2007.10129236

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. *BMC Med Genomics*, *4*, 31. https://doi.org/10.1186/1755-8794-4-31

Duke, M., Braden, R., Eddy, W., & Blanton, E. (2006). *RFC 4614: A roadmap for Transmission Control Protocol* (*TCP*) *specification documents*. RFC Editor, USA. https://doi.org/10.17487/RFC4614

FreeBSD Community. (2005). *Nfdump reference guide*. https://www.freebsd.org/cgi/man.cgi?query=nfdump&sektion=1&manpath=FreeBSD+8.2-RELEASE+and+Ports

Jin, D. D., & Lin, S. S. (2011). *Advances in computer science, intelligent systems and environment*. Springer. https://doi.org/10.1007/978-3-642-23777-5

Raja, M. S. N., & Vasudevan, A. R. (2017). Rule generation for TCP SYN flood attack in SIEM environment. *Procedia Computer Science*, *115*, 580–587. https://doi.org/10.1016/j.procs.2017.09.117

Sahi, A., Lai, D., Li, Y., & Diykh, M. (2017). An efficient DDoS TCP flood attack detection and prevention system in a cloud environment. *IEEE Access*, *5*, 6036–6048. https://doi.org/10.1109/ACCESS.2017.2688460

Samtani, S., Yu, S., Zhu, H., Patton, M., & Chen, H. (2019). Identifying SCADA vulnerabilities using passive and active vulnerability assessment techniques. In *2016 IEEE Conference on Intelligence and Security Informatics* (*ISI*) (pp. 25–30). Tucson, USA. https://doi.org/10.1109/ISI.2016.7745438

Sophos. (2019). *SophosLabs threat intelligence*. https://www.sophos.com/en-us/medialibrary/pdfs/factsheets/oem-solutions/sophos-threat-intelligence-dsna.pdf

Szmit, M. M., Wezyk, R., Skowroński, M. M., & Szmit, A. A. (2007). Traffic anomaly detection with snort. In *Information systems and computer communication networks*. Wydawnictwo Politechniki Wrocławskiej, Wrocław.

Zeigermann, L. (2016). *TIMESTAMP: Stata module to obtain a UNIX timestamp and the current time of a user-specified timezone*. https://EconPapers.repec.org/RePEc:boc:bocode:s458182

PERTEKLINIŲ TCP SESIJŲ SUDARYMO ATAKŲ ANALIZAVIMAS NAUDOJANT „NETFLOW“

V. Kapustin, N. Paulauskas

Santrauka

Srauto analizė – vienas pagrindinių įrankių anomalijoms kompiuteriniame tinkle aptikti. Atakos, konfigūracijos klaidos gali padėti lengviau pasiekti kompiuterinį tinklą ir galiausiai padidinti duomenų saugumo pavojų. Duomenų perdavimo tinklo srauto ir pavienių paketų analizė gali būti naudojama anomalijoms aptikti. Daugelis įrangos gamintojų įdiegia į savo įrangą srauto stebėjimo įrankius. „NetFlow“ protokolu perduodamu srautų duomenų kolektorius „Nfsen“ yra atvirojo kodo programinė įranga, padedanti surinkti informaciją iš agentų. Taip pat „Nfsen“ yra suprojektuota duomenų rinkinio įsibrovimo aptikimo sistemoms paruošti. Paruoštas duomenų rinkinys gali būti padalytas siekiant apmokyti ir testuoti dirbtinio intelekto modelį. Intelektinės sistemos srautui klasifikuoti gali būti naudojamas daugiasluoksnis perceptronas. Šiame darbe siekiama išanalizuoti, kaip interneto tiekėjo tinkle aptikti TCP perteklinį srautą ir jį klasifikuoti.

Reikšminiai žodžiai: „NetFlow“, tcpdump, TCP, paketas, ugniasienė, srautas, GRE, ataka.