USE OF MARKOV CHAIN FOR EARLY DETECTING DDOS ATTACKS

Chin-Ling Chen[1] and Jian-Ming Chen[2]

[1]Department of Information Management,
National Pingtung University, Pingtung, Taiwan 900
[2]Genesis Technology, Inc., HsinChu, Taiwan 300

ABSTRACT

DDoS has a variety of types of mixed attacks. Botnet attackers can chain different types of DDoS attacks to confuse cybersecurity defenders. In this article, the attack type can be represented as the state of the model. Considering the attack type, we use this model to calculate the final attack probability. The final attack probability is then converted into one prediction vector, and the incoming attacks can be detected early before IDS issues an alert. The experiment results have shown that the prediction model that can make multi-vector DDoS detection and analysis easier.

KEYWORDS

DDoS, attack detection, Markov chain, TCP SYN flood, ICMP flood, HTTP flood, LAND, UDP flood.

1. INTRODUCTION

In the context of Markov processes, Markov properties mean memory less which insinuates the random variables related to the future depend only on the related information at the current time, not that in the past. Several researchers have proposed to use the Markov model to assess the risk of security [1-5]. Karras et.al. (2008) have developed and evaluated a new security-related behavioural computing architecture and network model [1]. This model, based on Markov processes, calculates the key security factors related to intrusion detection and evaluation of security mechanisms. The calculation of the security parameters and the assessment of the security mechanism support the risk analysis and formulate the decision-making process under the balance of security and quality of service characteristics. Covert channels are used to transmit secret messages and spy on existing security equipment, posing a potential hazard to security. Therefore, it is necessary to develop a communication algorithm for the communicator. Zhai et.al. (2010) have analysed the potential redundancy of the TCP protocol [2]. According to the TCP protocol, the covert channel algorithm is categorized into many classes. Based on the TCP Markov model, this paper proposes a new covert channel detection method for different applications.

The Attack Tree Analysis (ATA) and Markov models are used for evaluation of the safety and availability of Building Automation Systems (BAS) over the life cycle. By analysing the probability of system weaknesses occurring in each cycle, ATA is applied to detect intrusions into BAS. The Markov model is used to calculate the feasibility of BAS, considering recovery and various error possibilities [3] (Abdulmunem and Kharchenko; 2016). The occurrence of vulnerabilities and subsequent attacks can cause the server to crash. In order to evaluate the availability and the functionality of server after DDoS attacks, Kolisnyk et.al. (2019) have

DOI: 10.5121/ijnsa.2021.13401

proposed a Markov model, in which the impact of successful DDOS attacks on some component and the transition rate of attacking between components have been evaluated [4].

The feature of the Markov chain is also memoryless. The transition probability *ij* depends only on the state *i* and not on the time *t* or not on the sequence of transitions happens before the time *t*. Cao (2007) has proposed an intrusion prediction technology based on Markov chain [6]. This predictive model uses a dynamic load balancing algorithm that avoids packet loss and false negatives in high-performance and high-load networks. Some of the security measurements rely on mathematical models, but most of them are based on practical data, qualitative methods and normative checks, leading to far-reaching results. Le and Hoang (2018) have proposed a method based on Markov chain and CVSS to calculate the probability distribution of cloud security threats [7].

Markov decision processes (MDP) extend Markov chain, mainly in choice and motivation. The action allows choices by decision maker, and rewards provide motivation for action. Wang et.al. (2016) have constructed a network security model that combines the MDP of the game model with the goal of summing up discount awards as well as solving individual nonlinear programming as a balancing strategy [8]. Some examples are used to calculate and apply sensitivity analysis to verify the feasibility and effectiveness of this method. MDP is a discrete-time, stochastic control process, used to deal with some results that may be partially random and partially optimized under the command of the decision maker. Miehling et.al. (2017) have presented a dynamic network security defense model using partially observable MDP [9]. This model is based on a dependency graph to construct the interaction between security conditions and exploits. To reduce the persistence of attacks, this model infers the security of the system and lists effective defense decisions. In MDP, the states can be changed randomly according to the action choices made by the decision-maker. MDP uses rewards to guide the action in the desired direction. Zheng and Namin (2018) have proposed a defense strategy for DDoS attacks on IoT devices with controlling network traffic optimization using MDP in an SDN environment [10]. The state of the environment influences the immediate rewards the agent has received and the future state transition probability. This model aims to choose actions that maximize a long-term outcome measure of total reward.

Most of the existing network security prediction mechanisms are only dominated by the prediction of state values, but these methods cannot reveal the problems caused by the dynamic characteristics of the network state factor.

In order to improve the correctness and time validity of network security status prediction, Kuang et.al. (2012) have proposed a Markov-based network security status prediction using the fuzzy method [11]. Based on the Markov state transition matrix, this method describes the association of network security, and predicts network status. The predicted value of the network security status is obtained by introducing vulnerability information to construct a fuzzy membership level, and integrating the improved Zadeh formula. Sun (2015) has proposed a network security state prediction mechanism based on complex network [12]. The dynamic characteristics of the oscillating value in the network security state prediction can be easily and intuitively tracked. This experiment result has shown that a Markov prediction method based on the complex networks can achieve an effective network security state prediction.

Zhou et.al. (2015) have proposed a multimodal-based anomaly detection mechanism to overcome the shortcomings of traditional anomaly detection [13]. In the mechanism, a classifier based on the intelligent Hidden Markov model (HMM) is designed to distinguish between attacks and faults. Based on an integrated simulation platform, an optimized performance network tool is used to analyse the accuracy and real-time performance of detection. Teoh et.al. (2016) have proposed the probability model of HMM to predict the characteristics of time series data, and constructed an expert system to predict the security attack [14]. This expert system, by parsing the historical record file, extracts the attacker's IP address and generates statistics, and divides the data into three groups: attack, uncertainty, and normal. The attributes are weighted to form a scoring system. Holgado et.al. (2020) have proposed a predicted model based on HMM using IDS alerts to forecast multi-step attacks [15]. Hidden states can be regarded as one stage of some special attack. In this model, the attack is adapted into a multi-step attack and the next step of the attack can be predicted with an offline training stage based on observations. The observations information in the training stage includes the mean number of alerts and the number of alerts in progress. Once the training is completed and the probability matrix is calculated, the predicted model uses both of the Viterbi and forward-backward algorithms to calculate the best state sequence based on the state probability of each step. The final attack probability is therefore obtained.

The rest of paper can be organized as follows. Section 2 describes the proposed scheme and discusses the early detection process. In Section 3, simulation and results are presented. Finally, we draw our conclusions in Section 4.

2. THE PROPOSED SCHEME

Suppose the attacker commands botnet to transmit *n* types of attack traffic to the victim in a short time to exhaust the victim resources, thus making service not being accessible for other users. Consider attacking type is a discrete-time Markov chain with transition probabilities, where

$$p_{ij} = \mathrm{e}^{-\lambda} \sum_{n=0}^{j} C_n^i \, p^n q^{i-n} \frac{\lambda^{j-n}}{(j-n)!} \text{, where } p + q = 1 \ (0 < p < 1) \qquad (1)$$

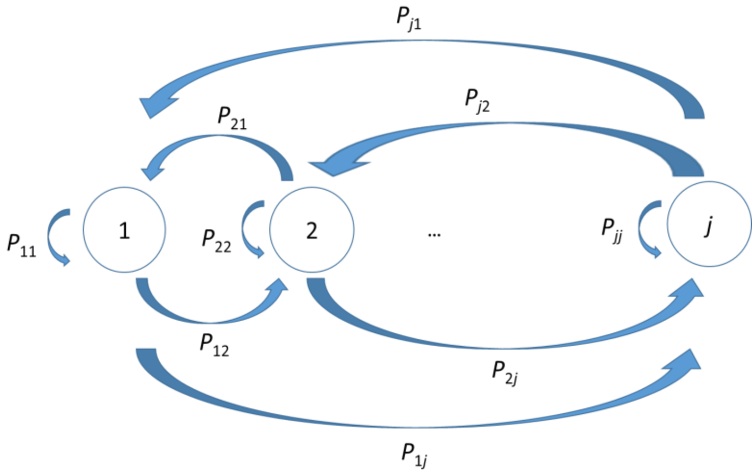


Figure 1. Markov Chain with Transition Probabilities

Let π_i be the equilibrium probability of attacking type *i*. We may write $\pi_i = \sum_{j=0}^{\infty} \pi_j p_{ji}$, where *j*= 0, 1, 2, ...

We make z-transform

$$
\begin{aligned}
P(z) &= \textstyle\sum_{i=0}^{\infty} \pi_i z^i \\
&= \textstyle\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \pi_i p_{ji} z^i \\
&= \textstyle\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \pi_i z^i \mathrm{e}^{-\lambda} \sum_{n=0}^{i} C_n^j \, p^n q^{j-n} \frac{\lambda^{i-n}}{(i-n)!} \qquad \text{(From (1))}
\end{aligned}
$$

Since $\sum_{i=0}^{\infty} \sum_{n=0}^{i} = \sum_{n=0}^{\infty} \sum_{i=n}^{\infty}$ we have

$$P(z) = \mathrm{e}^{-\lambda} \sum_{j=0}^{\infty} \pi_i \sum_{n=0}^{\infty} C_n^j p^n q^{j-n} \sum_{i=n}^{\infty} z^i \frac{\lambda^{i-n}}{(i-n)!}$$

$$= \mathrm{e}^{-\lambda} \sum_{j=0}^{\infty} \pi_i \sum_{n=0}^{\infty} C_n^j p^n q^{j-n} \sum_{i=n}^{\infty} z^{i-n} z^n \frac{\lambda^{i-n}}{(i-n)!}$$

$$= \mathrm{e}^{-\lambda} \textstyle\sum_{j=0}^{\infty} \pi_i \sum_{n=0}^{\infty} C_n^j p^n q^{j-n} \sum_{i=n}^{\infty} z^{i-n} z^n \frac{\lambda^{i-n}}{(i-n)!}$$

$$= \mathrm{e}^{-\lambda} \sum_{j=0}^{\infty} \pi_i \sum_{n=0}^{\infty} C_n^j p^n z^n q^{j-n} \sum_{i=n}^{\infty} z^{i-n} \frac{\lambda^{i-n}}{(i-n)!}$$

$$= \mathrm{e}^{-\lambda} \mathrm{e}^{\lambda z} \sum_{j=0}^{\infty} \pi_i \sum_{n=0}^{\infty} C_n^j (pz)^n q^{j-n}$$

Since $C_n^j = 0$, for $n>j$, then

$$P(z) = \mathrm{e}^{\lambda(z-1)} \textstyle\sum_{j=0}^{\infty} \pi_i (pz+q)^j$$

Therefore,

$$P(z) = \mathrm{e}^{\lambda(z-1)} P[1 + p(z-1)]$$

P(*z*) is the final probability in given model. There are four QoS parameters in the proposed model: CPU load, response time, total processes and packets lost. We use one vector store the final probability of four QoS parameters.

P = ($P_c(z)$, $P_r(z)$, $P_p(z)$, $P_l(z)$), where $P_c(z)$, $P_r(z)$, $P_p(z)$ and $P_l(z)$ represent the final probability of CPU load, response time, total processes and packets lost, respectively.

3. Experiment and Results

We setup a testbed scenario for experiment (Fig.2). Nagios (monitor) periodically queries the victim. The return service status can be OK, warning or critical. We illustrate four QoS parameters: CPU load, total processes, response time and packets lost. We use five DDoS attack scenarios to demonstrate the feasibility of the prediction system. They are TCP SYN flood, ICMP flood, HTTP flood, LAND and UDP flood. We use LOIC DDoS generator to generate TCP SYN flood, UDP flood and HTTP flood attack, and use Hping3 to produce ICMP flood and LAND attack.

3.1. TCP SYN flood attack

In Fig.3, Load 15, Load 5 and Load 1 indicate the CPU load of latest 15 min, 5 min and 1 min, respectively. Last, Average, Max represents the last, average and maximum CPU load during the time period, individually. In this scenario, we use TCP SYN flood as the attack source. The attacker sends thousands of SYN packets using a fake IP address to every possible port at the victim server. The victim server responds to the innocent client at fake IP address with a SYN-ACK packet. The innocent client, of course, does not send back the expected ACK. The victim server will continue to monitor the open ports to wait for ACK of its corresponding SYN-ACK packet for some time, thus spending some CPU execution. During the waiting time, either the connections stay open until timeout or the victim server sends an RST packet to close the connection. Both make the response time of connection completion increase. Nagios sends the critical alert of CPU load (Fig.3) and response time (Fig.4) at 21:05, individually. However, the prediction system indicates the imminent attack at 20:55 (Fig.3).

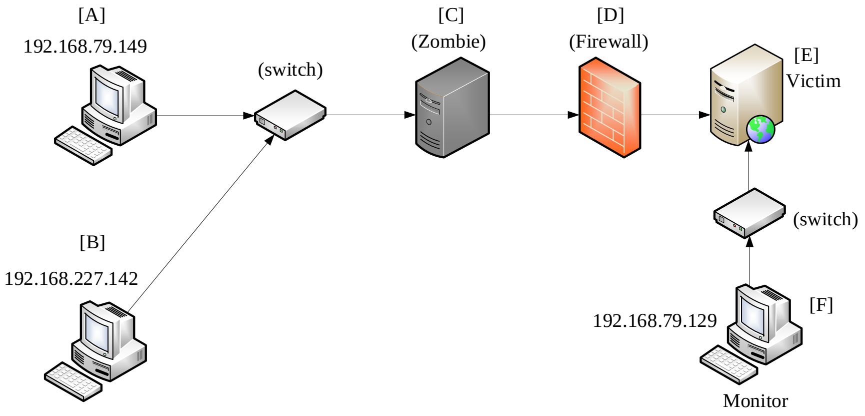
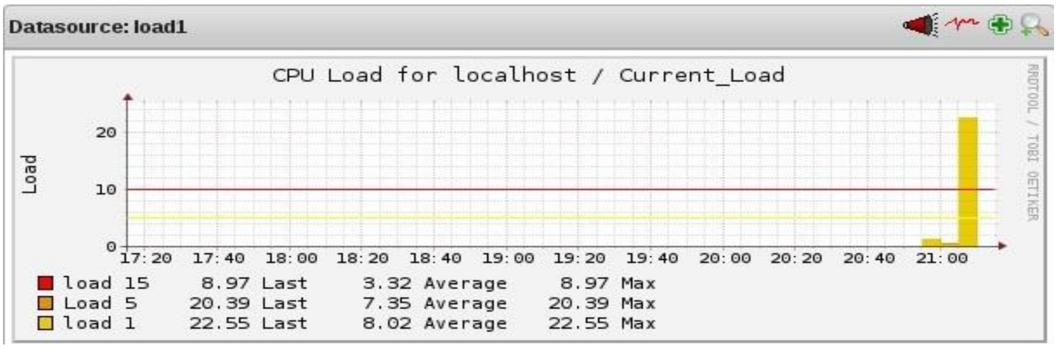
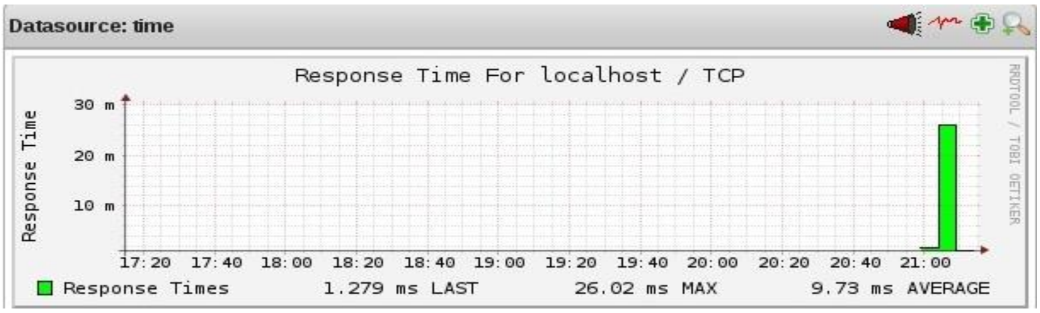


Figure 2. Testbed Scenario



```
🟥 load 15    8.97 Last    3.32 Average    8.97 Max
🟧 Load 5    20.39 Last    7.35 Average   20.39 Max
🟨 load 1    22.55 Last    8.02 Average   22.55 Max
```

Figure 3. CPU Load under TCP SYN Flood Attack



```
🟩 Response Times        1.279 ms LAST        26.02 ms MAX        9.73 ms AVERAGE
```

Figure 4. Response Time under TCP SYN Flood Attack

3.2. ICMP flood attack

In this scenario, we use the ICMP flood (Ping flood) as the attack source. Nagios only sends the warning alert of CPU load (Fig.5) at 22:35. From that time on, ICMP flood attack does not incur increasing CPU load. The reason is that the size of ping requests is small, which has little impact on CPU execution. However, Nagios sends the critical alert of packets lost (Fig.6) and response time (Fig.7) at 22:32 and 22:45, individually. ICMP flood put a strain on the channel of the network, consuming significant bandwidth and resulting in a great number of packets lost. An eminent packet loss may worse the response time and lead to notable delays. Nevertheless, the prediction system indicates the imminent attack at 22:21 (Fig.5).

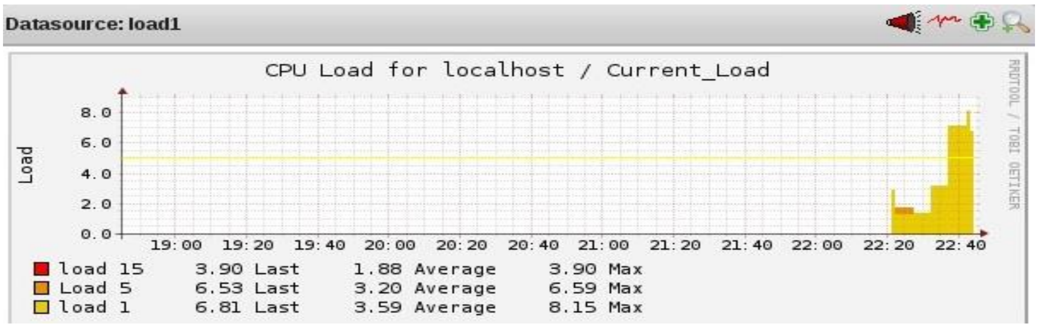


Figure 5. CPU Load under ICMP Flood Attack

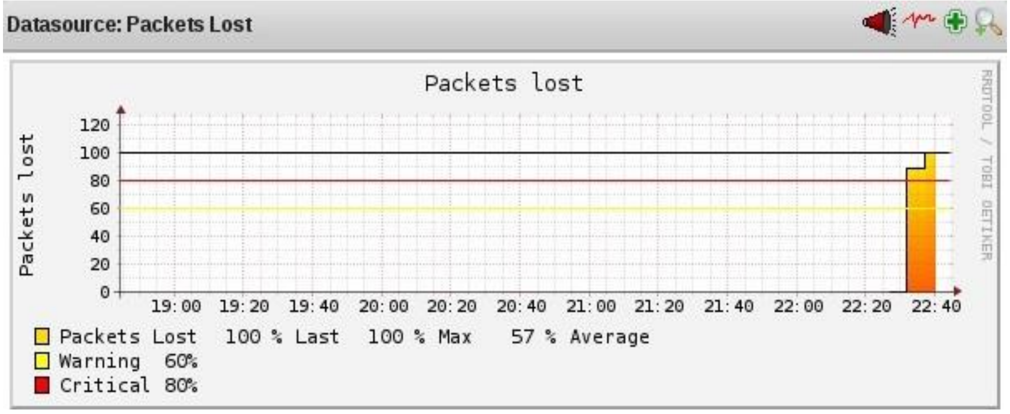


Figure 6. Packets Lost under ICMP Flood Attack

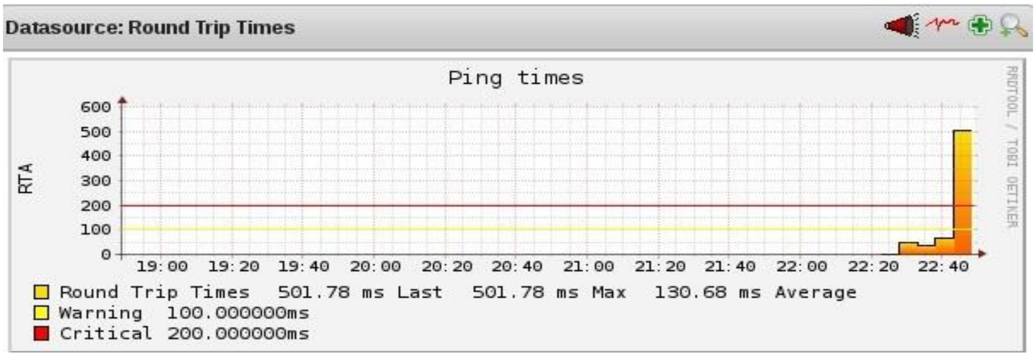


Figure 7. Response Time under ICMP Flood Attack

3.3. HTTP flood attack

In this scenario, we use HTTP flood as the attack source. HTTP flood tries to overwhelm the victim server by sending seemingly-legal HTTP GET or POST requests. However, Nagios only shows the warning alert of total processes (Fig.8). HTTP flood attack forces the victim server to allocate the resources as much as possible in response to each HTTP GET or POST request, and each of them is treated as a processing-intensive job. Therefore, Nagios sends the critical alert of CPU load (Fig.9) at 23:52. HTTP floods give rise to complex server-side processing, and require less bandwidth than other attacks to submerge the victim server. Traditional volume-based detection is hard to observe HTTP flood because the incoming attack traffic is always under the threshold. As we mentioned earlier at section 3.2, the amount of bandwidth available is correlated to response time. The response time seems acceptable even under the attack of HTTP flood (Fig.10).

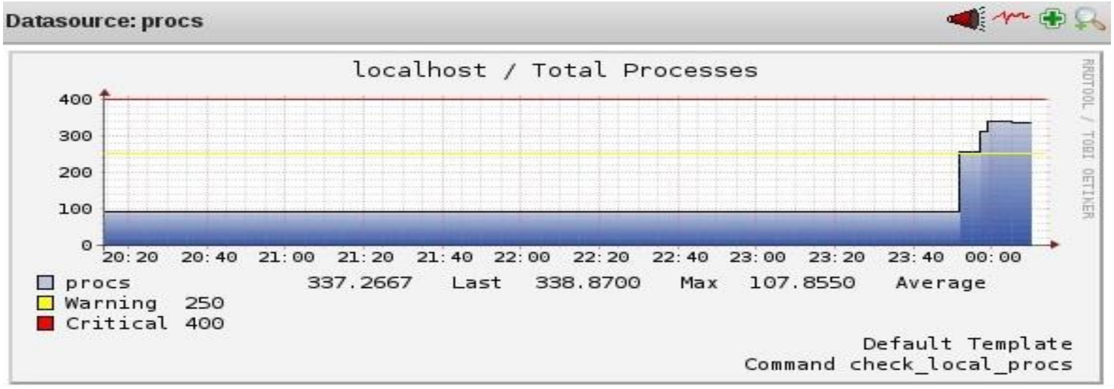


Figure 8. Total Processes under HTTP Flood Attack

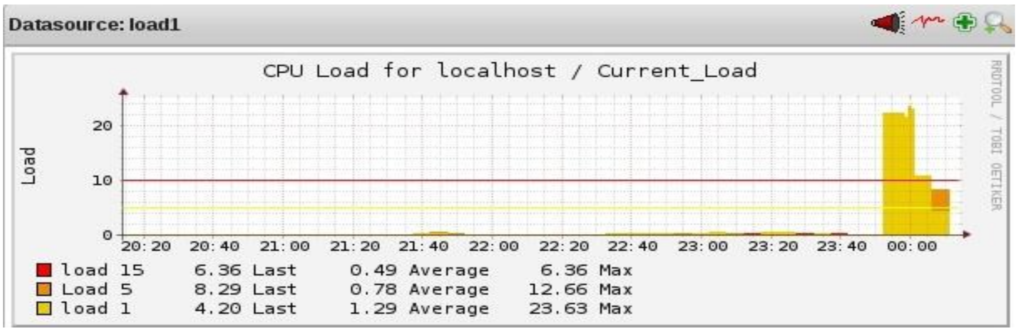


Figure 9. CPU Load under HTTP Flood Attack

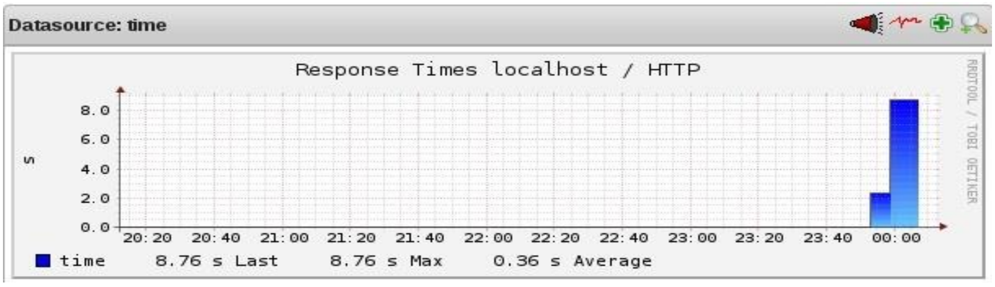
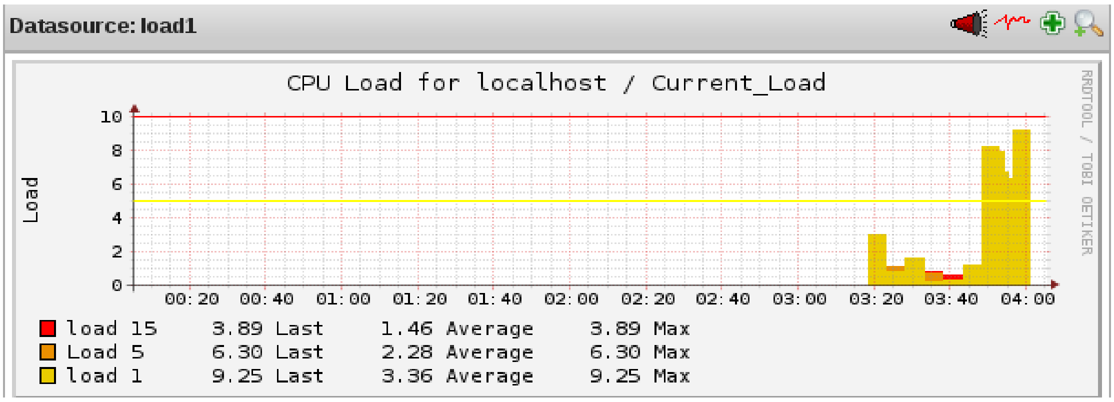


Figure 10. Response Time under HTTP Flood Attack

3.4. LAND attack

In a LAND (Local Area Network Denial) attack, the attackers use the IP spoofing technique to create a crafted TCP packet which contains the same source address (port) and destination address (port). A victim server receiving such crafted packets replies to itself, thus causing an increase of CPU load. Like other state-of-art IDS, Nagios shows lower detection (warning) of performance of CPU load (Fig.11), total processes (Fig.12) and response time (Fig.13). However, the prediction system can detect LAND attack earlier (03:20) based on multi-variable technique.

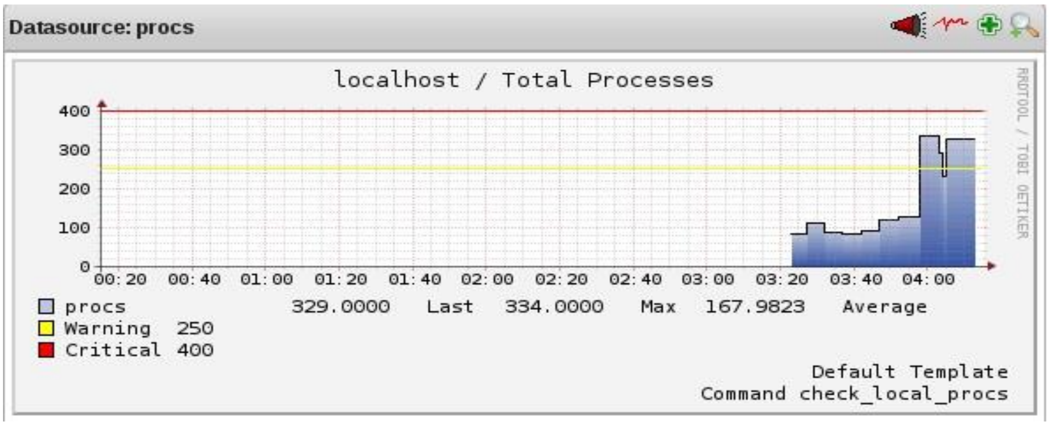


Datasource: load1

CPU Load for localhost / Current_Load

```
🟥 load 15    3.89 Last    1.46 Average    3.89 Max
🟧 Load 5     6.30 Last    2.28 Average    6.30 Max
🟨 load 1     9.25 Last    3.36 Average    9.25 Max
```

Figure 11. CPU Load under LAND Attack



Datasource: procs

localhost / Total Processes

```
🟦 procs              329.0000    Last    334.0000    Max    167.9823    Average
🟨 Warning   250
🟥 Critical  400
                                                          Default Template
                                                  Command check_local_procs
```

Figure 12. Total Processes under LAND Attack

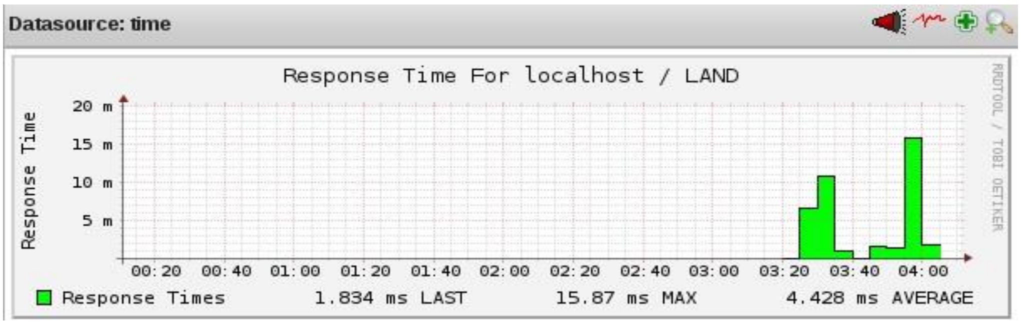


Figure 13. Response Time under LAND Attack

3.5. UDP flood attack

In this scenario, we use the UDP flood as an attacking source. Without a three-way handshake like TCP, UDP traffic runs with lower overhead and requires less resources. Nagios only shows the warning alert of CPU load (Fig.14). UDP does not define specific packet formats. Therefore, the hackers can produce large packets and fill the crafted packets with junk messages at his own. UDP flood generates a high volume of UDP junk traffic to swamp the victim server. Just as mentioned at section 3.2, the remaining of bandwidth available has impact on the response time. The response time grows as bandwidth consumption increases. Nagios sends the initial critical alert of response time (Fig.15) at 03:20.

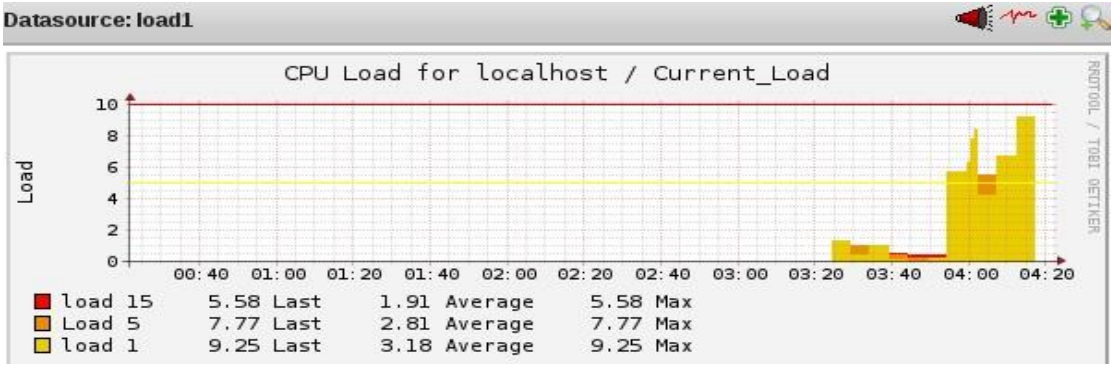


Figure 14. CPU Load under UDP Flood Attack

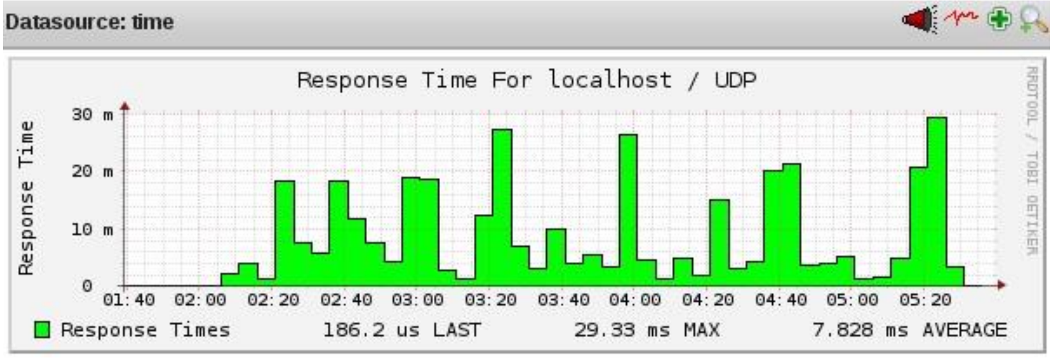


Figure 15. Response Time under UDP Flood Attack

4. CONCLUSIONS

DDoS attacks are arduous to detect because they can generate the attack traffic to flood the victim host from a large number of zombie machines. In this paper, we have designed and implemented an early attack detection mechanism that adopts the final probabilities to recognize the incoming attack. We have set up Nagios based on a Linux-based platform to monitor all mission-critical infrastructure components including services, operating systems and network protocols. Nagios is configured to forward warning alerts whenever there is possible malicious traffic. The experiment results have shown that the proposed mechanism can detect incoming DDoS attacks early as well as effectively.

REFERENCES

[1] Karras, D. A. &Zorkadis,V. C. (2008) "On efficient security modelling of complex interconnected communication systems based on Markov Processes," 2008 New Technologies, Mobility and Security, pp1-7.

[2] Zhai, J., Liu, G.& Dai, Y. (2010) "A covert channel detection algorithm based on TCP Markov model," 2010 International Conference on Multimedia Information Networking and Security, pp893-897.

[3] Abdulmunem, A.-S. M. Q.&Kharchenko, V. S. (2016) "Availability and security assessment of smart building automation systems: combining of attack tree analysis and Markov models," 2016 Third International Conference on Mathematics and Computers in Sciences and in Industry (MCSI), pp302-307.

[4] Kolisnyk, M.,Kharchenko, V.&Iryna, P. (2019) "IoTserver availability consideringDDoS-attacks: analysis of prevention methods and Markov model," 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT).

[5] Shing, M. -L.&Shing, C. -C. (2010) "Information security risk assessment using Markov models," 2010 Third International Symposium on Electronic Commerce and Security, pp403-406.

[6] Cao,L. -C. (2007) "A high-efficiency intrusion prediction technology based on Markov Chain," 2007 International Conference on Computational Intelligence and Security Workshops (CISW 2007), pp518-521.

[7] Le,N. T.& Hoang,D. B. (2018) "Security threat probability computation using Markov Chain and common vulnerability scoring system," 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), pp1-6.

[8] Wang, C., Shi, C., Wang, C.& Fu,Y. (2016) "An analyzing method for computer network security based on Markov game model," 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp454-458.

[9] Miehling, E., Rasouli, M.&Teneketzis, D. (2017) "A dependency graph formalism for the dynamic defense of cyber networks," 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp511-512.

[10] Zheng, J.&Namin,A. S. (2018) "Defending SDN-based IoTnetworks againstDDoSattacks using Markov Decision Process," 2018 IEEE International Conference on Big Data (Big Data), pp4589-4592.

[11] Kuang, G. C., Wang, X. F.& Yin, L. R. (2012) "A fuzzy forecast method for network security situation based on Markov," 2012 International Conference on Computer Science and Information Processing (CSIP), pp785-789.

[12] Sun, S. (2015) "The research of the network security situation prediction mechanism based on the complex network," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), pp1183-1187.

[13] C. Zhou, S. Huang, N. Xiong, S. -H. Yang, H. Li, Y. Qin & X. Li, (2015) "Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2015, Vol.45, No.10, pp1345-1360.

[14] Teoh, T. T., Nguwi, Y. Y., Elovici, Y., Cheung, N. M.& Ng, W. L. (2017) "Analyst intuition based Hidden Markov Model on high speed, temporal cyber security big data," 2017 13th International

Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pp2080-2083.

[15] Holgado, P., Villagrá, V. A.& Vázquez, L. (2020) "Real-time multistep attack prediction based on Hidden Markov Models," *IEEE Transactions on Dependable and Secure Computing*, 2020, Vol.17, No.1.

AUTHORS

Chin-Ling Chen received the BS degree from National Taiwan University in 1988, the Master degree in Management Information System from the University of Wisconsin, Milwaukee, in 1992, and the Ph.D. degree in Information Management from National Taiwan University of Science and Technology, 1999. Since the spring of 1999, he has joined the faculty of the Department of Information Management at National Pingtung University, Taiwan. His research interests include Internet QoS, network technology, and network security. He is a member of IEICE.

Jian-Ming Chen received his master's degree in Information Management from National Pingtung University, 2019. Currently, he is a software engineer of Genesis Technology, Inc, Hsinchu, Taiwan.