American Journal of Applied Sciences 9 (6): 909-916, 2012
ISSN 1546-9239

A Discrete Event Simulator for Extensive Defense Mechanism for Denial of Service Attacks Analysis

[1]Maryam Tanha, [1]Seyed Dawood Sajjadi Torshizi and [2]S. Shamala
[1]Department of Computer and Communication Systems Engineering,
Faculty of Engineering,
[2]Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Abstract: Problem statement: Seeking for defense mechanisms against low rate Denial of Service (DoS) attacks as a new generation of DoS attacks has received special attention during recent years. As a decisive factor, evaluating the performance of the offered mitigation techniques based on different metrics for determining the viability and ability of these countermeasures requires more research. **Approach:** The development of a new generalized discrete event simulator has been deliberated in detail. The research conducted places high emphasis on the benefits of creating a customized discrete event simulator for the analysis of security and in particular the DoS attacks. The simulator possesses a niche in terms of the small scale, low execution time, portability and ease of use. The attributes and mechanism of the developed simulator is complemented with the proposed framework. **Results:** The simulator has been extensively evaluated and has proven to provide an ideal tool for the analysis and exploration of DoS attacks. In-depth analysis is enabled by this simulator for creating multitudes of defense mechanisms against HTTP low rate DoS attacks. The acquired results from the simulation tool have been compared against a simulator from the same domain. Subsequently, it enables the validation of developed simulator utilizing selected performance metrics including mean in-system time, average delay and average buffer size. **Conclusion:** The proposed simulator serves as an efficient and scalable performance analysis tool for the analysis of HTTP low rate DoS attack defense mechanism. Future work can encompass the development of discrete event simulators for analysis of other security issues such as Intrusion Detection Systems.

Key words: HTTP low rate DoS, DES, mean in-system time

INTRODUCTION

Denial of Service (DoS) attacks is considered as one of the most high-profile security threats to the network and communication systems. It has a devastating impact on many crucial online businesses especially electronic banking and e-Commerce services as well as compromises political websites (SOPHOS, 2011; Ghazali and Hassan, 2011; Rhazi *et al*., 2007; Viswanathan *et al*., 2012). This impact has contributed to the rich surge of research concerning the discovery of mitigation techniques to defend systems against these types of attacks. In DoS attacks, intruders attempt to abuse preliminary properties of communication systems. For example, they attempt to send huge number of requests to a destined target. Thus, creating an overflow in its buffer (i.e., queue). This is one of the most widespread methods to launch a DoS attack against a target in any kind of network, especially over the Internet (Luo and Shyu, 2005; Jensen *et al*., 2008). Constantly, many web sites fall victims of this type of attack in the Internet. However, the generation of the enormous amount of traffic to saturate the server's resources is one of the primary problems faced by the attackers and subsequently creates a substantial hindrance. Typical of an attacker's behavior, a new generation of DoS attacks has come into existence surpassing these hindrances. These attacks have mostly aimed at the application layer as compared to the other types of existing DoS attacks (i.e., SYN floods). Subsequently, compromising the vulnerabilities in the network layer of the TCP/IP model.

Corresponding Author: S. Shamala, Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology, University Putra Malaysia,
43400 Serdang, Selangor, Malaysia Tel: +60389471748 Fax: +603-89466576

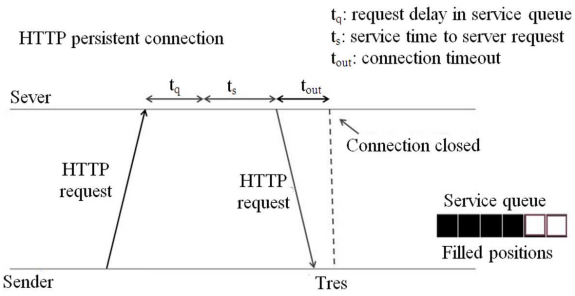

Fig. 1: Function of HTTP/1.1 persistent connection

Furthermore, these new attacks have displayed attributes which require reduced amount of resources (e.g., bandwidth) for launching an attack, making it highly effective and is also more difficult to detect (Guirguis *et al*., 2006).

Kuzmanovic and Knightly (2006), the authors introduced a type of DoS attacks known as "Low-rate TCP-targeted DoS attack" that unlike the mass types of DoS attacks does not generate a high magnitude of traffic. In this attack type, a single user can interfere the norm functions of a network device easily even with low bandwidth Internet connectivity. This idea was further abused to make new generation of web attacks against web servers which serve diverse services over Internet. Hackers and intruders attempted to make use of this technique to interfere the operations of HTTP web servers especially by misusing the rudimentary characteristics of HTTP persistent connections in the HTTP/1.1 protocol. The attackers attempt to keep the service queue of the HTTP server constantly in a saturated condition by sending scheduled HTTP requests that consume the network resources, subsequently preventing valid web users to connect to the server as well as served by the targeted web server. Fig. 1, shows the functionality of a HTTP persistent connection that upon the establishment of a HTTP connection, the HTTP request remains in the service queue for a total of t_q+t_s seconds to be served by the HTTP server. After the generation of the HTTP response by the web server, the established connection will remain as an open status for a total time of t_{out} seconds to serve other requests from the same user who have established the HTTP connection for first time. The completion of the $t_{out,}$ period, the established connection will be terminated and seized its respective position in the HTTP service queue will be released to be consumed by other web users (Macia-Fernandez *et al*., 2008).

The need for more comprehensive solutions for identifying these security threats has created the constant urge for more research in this area. In addition, performance evaluation of the offered solutions based on different metrics is among the most imperative factors to determine the feasibility and ability of these countermeasures. Considering the various conditions and dimensions of evaluation of the multitudes of the proposed and developed solutions will inevitably increase the complexities of any form of performance analysis. The prevailing DES simulators such as the Network Simulator 2 (NS-2) have catered these challenges but with equally demanding attributes such as complexity, portability and resource consumption. An alternate solution to confine these demands, while furnishing all the imposed pre-requisites of an analysis is proposed in this study. The development of a customized DES simulator enables the achievement of precise results whilst optimizing the related computing resources. The dichotomy of the proposed DES has enabled the ability for extensions towards other analysis.

Related work: The essence of the proposed simulator has originated from several factors. Among the main factors is the potential possessed by a general-purpose simulator with low memory and low processing consumption. This study analyzes in detail the features of existing performance analysis strategies and tools for DoS attacks with special attention placed on a comparative angle.

There has been a wide range of research on performance analysis tools for DoS attacks which are dominant in different fields of communication and networking. Jin *et al*. (2009) the authors studied the applications of simulation analysis on DoS attacks and defense mechanisms. They analyzed and simulated the processes of DoS attacks and the defense techniques. The research eventually proposed an enhanced design for simulating DoS defense strategies via the usage of NS-2. Luo and Shyu (2005) simulated a framework to maintain QoS of the multimedia streams during the DoS attacks by using NS2 also. The acquired simulation results showed the effectiveness of the offered framework with regards to the buffer occupancy and the Peak Signal-to-Noise Ratio (PSNR) values of the received video. The simulation of low rate DoS attacks was investigated in (Zhu *et al*., 2011). They analyzed low rate DoS attacks using frequency domain since these attack flows are periodic and it is tough to observe and detect such attacks in time domain. In addition, they employed parameters such as channel utilization ratio, packet loss ratio and average congestion window to study the behavior of the system. Shevtekar and Ansari (2006), the research encompassed the simulated effects of low rate DoS attacks on VoIP QoS sensitive traffic in terms of delay, jitter and packet loss.

Table 1: Comparison between four simulators

Feature	NS2	OMNet++	QualNet	OPNET
Ease of use (GUI support)	Poor	Excellent	Satisfactory	Excellent
Cost of licenses	Excellent (open source and free)	only free for academic and non-profit use	Poor (Relatively expensive)	Poor (Relatively expensive)
Extensibility	Excellent	Excellent	Excellent	Excellent
Platform	Linux, Unix- based operating systems, possible to be installed on Windows	Linux, Unix-like systems, Mac OS X, Windows (XP, Win2K, Vista, 7).	Linux, Unix-like systems, Mac OS X, Windows (XP, Vista, 7)	Linux, Windows (XP,Vista,7), Windows Server 2003, 2008
Documentation	Satisfactory	Satisfactory	Satisfactory	Excellent

By using NS-2 simulator to perform some scenarios, it was demonstrated that the packet loss was considerably affected by increasing burst period or burst rate. Thus, resulting in the reduction of quality or complete denial of service of VoIP calls. Lin *et al.* (2008), a simulated analysis on a queue-based scheme was proposed to reduce malicious packet flow of distributed DoS attacks. In both of the former and latter cases, researchers have used NS2 as a discrete event simulator tool to evaluate their presented models.

Gabriel Macia Fernandez and his team have published multiple papers about HTTP low rate DoS attacks and also proposed several countermeasures to defend target servers against these attacks (Macia-Fernandez *et al*., 2006; 2008; 2010). Their research used multiple performance metrics such as mean in-system time and attack efficiency to evaluate the functionality of offered solutions in NS2 software and Linux test bed environment.

As stated in (Reineck, 2008), selecting a specifically tailored simulator is indeed challenging due to the availability of many tools as well as the time-consuming process involved in the familiarization of the simulator and its respective functionalities. Thus usually a researcher picks one tool and keeps using it for all his/her research or some opt to develop his/her customized simulator utilizing general purpose languages. A detail comparison has been done between four dominant and successful discrete event simulators, namely the NS2, OMNeT++, QualNet and OPNET. The analysis has been presented in a Tabular form (i.e., Table 1) from the review and analysis of (Reineck, 2008; Begg *et al*., 2006; OMNeT++, 2001-2009; SNT, 2008-2012; OPNET, 2012).

Upon analyzing these simulators, NS2 is considered as a preferred better choice and a strong tool with regard to cost-effectiveness, extensibility, widespread use and extend of usage. Moreover, the results of NS2 are able to resemble the test-bed results with close proximity (Begg *et al*., 2006). Although it is weak in the aspect of ease of use (i.e., the user have fundamental knowledge regarding how to work with Linux-based operating systems, installing required packages, package management and package dependencies, even in some cases upgrading the GNU C Compiler and so on) but its benefits outweighs these disadvantages. The proposed and developed simulator in this research has embarked in providing precision of results as those achieved by NS2, whilst striking an ideal balance of ease of use. Then presents the proposed simulator.

MATERIALS AND METHODS

The developed taxonomy in Table 1 has served as the underlying design factor for the proposed simulator. The general consensus is suitable modeling and analysis techniques will be determined based on the attributes and characteristics of a particular system. The essence of understanding the mechanics of a system operation is to capture it in a model and define all the possible inter-relationship at the determined level of abstractions. Among known modeling methodologies, discrete-time event-driven models have positioned themselves within the dominant performance analysis tools and encompass a wide spectrum of computer and communication systems. In Discrete Event Simulation (DES), the function of system is captured and reflected as a chronological sequence of events; where each event occurs at an instant in time and marks a change of state in the system (Robinson, 2004). Relating the core principles of the DES to current and evolving research areas requires precision mapping and the tailoring of the area to the queuing theory norms. This mapping imposes onto the analyst the need to relate multiple levels of abstractions into the appropriate queuing paradigm. Network security is among the areas of research rich in potential and is paving its way to wider horizons. This study engages these potentials from an analysis angle. The rich research potential has been ventured by many performance analysis tools as deliberated. This research has developed a DES with a specialized purpose. Parallel to the development of a tool and of equal importance magnitude is the

verification of the tool and its relevance to the fraternity. This research work has selected the verification technique which utilizes as established work as a comparative benchmark.

Thus, the study of (Macia-Fernandez *et al*., 2010) which utilized extensively the simulation technique was chosen among the existing performance evaluation techniques. The comprehensive and powerful discrete event NS2, enabled the proposed model for the system to be simulated. The simulation results in this work had been validated by implementing a test-bed for HTTP low rate DoS attacks. Subsequently, the acquired results from NS2 were proven to be coherent with those acquired from real environment. Prior to embarking on the details of the proposed and developed simulator in this study, a detail description of the underlying algorithms are deliberated. This is done in line with (Macia-Fernandez *et al*., 2010) to ensure that a level playing field is achieved in forming the relevance of this research.

HTTP server model in low rate DoS: The developed model for the HTTP application server in (Macia-Fernandez *et al*., 2010) is depicted in Fig. 2.

The server consists of a service queue for incoming requests (attacker's request or legitimate user's request) and one or more service modules responsible for processing requests. The Queue discipline imposed can be of any nature such as the First Come First Serve (FCFS), Weighted Fair Queuing (WFQ). When a service module completes processing the request, the respective response is sent to the corresponding client and the service module becomes idle. Producing the response will eventually result in a free position in the queue. Thus, either an attacker's request or a valid user's request will be able to seize a position in queue. A HTTP low rate DoS attack endeavors to impede legitimate users from occupying free places in the service queue of the server by overwhelming the queue with requests. To achieve this goal, the attacker attempts to derive precise predictions as to when a free space will be available in the queue of server (i.e. by exploiting the HTTP persistent connection feature of the server) and generates a request within the predicted time. Then discussed the mechanism of defense against such attacks.

Random Service Time (RST) as a defense mechanism: Random Service Time (RST) (Macia-Fernandez *et al*., 2010) as a defense mechanism designed to make it less viable to predict the server behavior by eradicating fixed patterns from the server's operation. RST attempts to randomize the constant timeout of server so that the attacker will not be able to foresee the answer instantly. Therefore, when the service is ended by the service module, the module stays locked for an extra time denoted as Δt_{RST} (Fig. 3). Incorporating the Δt_{RST}, in the Round Trip Time (RTT) computation, (i.e. the RTT being the period of time between the response (a.k.a. answer) acquisition instant (t_1 in Fig. 3) and the receiving of the attack packet sent as a response to the answer (t_2 in Fig. 3), an empty position present in the service queue can be occupied by a legitimate user (because the attacker doesn't know the value of $\Delta t_{RST)}$. The employed formulas for obtaining the Δt_{RST} are not in the scope of this study. A detailed explanation of this information on the attack implementation is provided in (Jin *et al*., 2009). In this research we considered multiple attack threads that share information pertaining to connections to the server. The attackers pool their data in a separate but common queue, called positions queue that is made available to all of them. In this way they are able to improve the efficiency of the attack and prevent sending attack traffic to the server with no benefit.

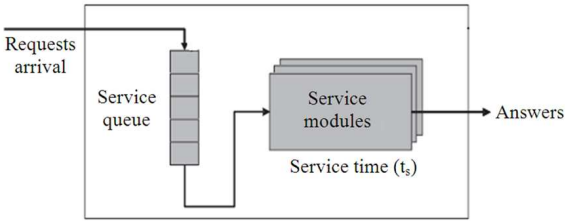

Fig. 2: Application server model (Macia-Fernandez *et al*., 2010)

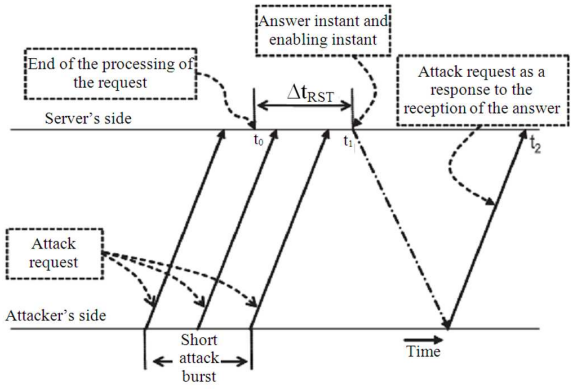

Fig. 3: HTTP low rate Dos attacks attack process when using RST (Macia-Fernandez *et al*., 2010)

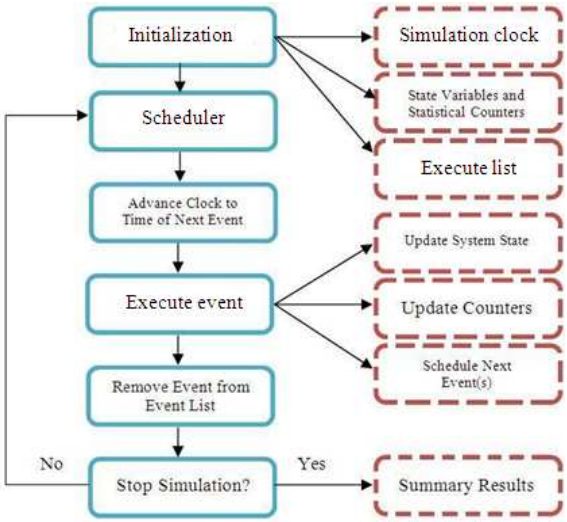

Fig. 4: Main event scheduling mechanism in DES

Proposed discrete event simulator for performance analysis of DoS: Figure 4 illustrates the main components associated to the structure of the event scheduling process in discrete event simulation. Upon the completion of the initialization of the system state variables and statistical counters, the scheduler selects the event with the least start time from event list to be executed and the clock will be advanced accordingly to the start time of the selected event. The execution of each event, the simulation related transactions will be complemented by the checking of the Termination of Simulation (ToS). If the requirements for ending the simulation are met, the results will be generated.

The essence of this research is embedded in the development of the DES as a pertinent performance analysis tool for the DoS represented in Fig. 2. The derivation of the problem into fundamental queuing models forms the basis of the developed simulator. The attacks which originate from numerous origins are derived to be represented by multiple sources. The common service queue is mapped on a single queue representation and the core resource as a single data sink model was developed. Variations to this model can be easily incorporated into the proposed simulator based on the multitudes of elements (i.e. 'm' data sinks and 'x' number of queues). Though, the mechanism of security modeled encompasses a wide gradient of computational activities, the approach adopted to form the modularity of the developed simulator is to reduce the distinction and form main clusters of events based on the impact created in a holistic manner. Thus, two types of events have been derived, which are the: request arrival and request answer. Arrival is derived to be either a legitimate user's request or an attacker's request. This is done based on the acquired results of the request impact onto the algorithm and its reactive features. The request differentiation will befit the attackers' effort to disguise as legitimate user. The further justification for choosing arrival and answer as events is that they cause an obvious and substantial change in the statistical composition of the system. In this research, the statistical composition is taken to be the number of requests. In complementing the logical component of the developed simulator with those of the simulator custodian, the time advancing approach was developed. It should be noted that assigning start time to events (i.e. the time an event is to occur) must ensure the reflection of the actual sequence. In this research, the commencement of the simulation is from time zero and does not incorporate a system already in operation. As such, it is obligatory for the first event to be an arrival then only the corresponding answer event is permitted to happen. The furnishing state variable is the state of the server which is derived to be either idle or busy. The performance metrics are utilized to characterize the algorithms / systems under study. The number of requests arrived and also the number of requests answered, are defined as the statistical measures that are used to both reflect the system ability but more importantly in this research as the verification tool for the developed simulator.

The request arrival generation of legitimate users is derived using the Poisson distribution. The C built-in random number generator was adopted to generate the random variable within the Poisson distribution and a sequence of independent arrival times. In the effort to make the comparative analysis of the developed simulator with an existing standard for the purpose of validation the following assumptions stated in (Macia-Fernandez *et al*., 2010) has been adopted:

- At the beginning of simulation, the queue is full of requests
- The queuing discipline is FCFS
- The Attacker sends a burst of traffic containing three attack requests
- Poisson distribution is used for generation of both types of request (i.e. attack request and legitimate user request)
- Request generation rate is fixed for legitimate users' requests
- Service time is considered fixed
- New incoming requests will be ignored (dropped) if the queue is full (packet discarding strategy)
- The length of queue is equal to the number of attack threads

```
int main()
{
// Max-Δt RST is equal to the number of experiments
for (int Δt RST = 0; exp_no < Max- Δt RST; ++Δt RST)
{
//new experimental execution without explicit interaction from the user, determining starting
values for events and state variables (e.g. setting the number of requests for each experiment
simclock and performance metrics to zero and so on)
init ();
// Here, ToS (Termination of Simulation) is by time
while (simclock < ToS)
{
// Choosing the event (i.e. arrival of a request or answer to a request) with the smallest
start time
scheduler();
// Showing event based time advancing mechanism
update_clock();
// Event activation (triggering a request arrival or a request answer)
switch (event_type)
{
//arrival of a legitimate user's request or an attacker's request
case EVENT_ARRIVAL:
arrival();
break;
// request's answer
case EVENT_ANSWER:
answer();
break;
}
// calculating average buffer size (i.e. average size of service queue)
buffer_management();
}// end of while
// Printing results for Kth Δt RST (i.e. Kth experiment)
results (Δt RST);
}// end of for
getch();
return 0;
}
```

Fig. 5: Main body of the simulation code

By developing a discrete event simulator from scratch, one can thoroughly dissect the functionalities of the mechanism under study as compared to utilizing simulation tools such as NS2. The pseudo-code of the proposed and developed simulator is shown in Fig. 5.

Performance metrics: The mean in-system time which is the total in-system time divided by number of requests answered is adopted from (Macia-Fernandez *et al*., 2010) as the performance metrics. In-system time is the time passed between the arrival of a request and the generation of its corresponding answer. Thus, it can also be adopted as the delay in service queue for a request plus the required service time. Mean in-system time is of great importance due to the fact that it is able to assist the process of judging the influence of RST as a defense mechanism on the ordinary behavior of the server. An ideal defense technique should minimize this impact.

In addition to mean in-system time, the two other performance metrics used include average delay and average queue size. Average delay is calculated is done by dividing the total delay by the number of requests answered. The average buffer (queue) size is the total queue size divided by simulation clock for each experiment as illustrated in Fig. 5 (i.e., the buffer_management function). This provisions that the queue occupancy at each discrete time segment is reflected by the area under the curve.

RESULTS AND DISCUSSION

The primary function of the acquiring the results is to utilize as a testimony of the simulator validity. This was done by incorporating the control parameters stated in Table 2 which is in tangent with (Macia-Fernandez *et al*., 2010).

Table 2: Values for parameters in system

Parameters	Values
Duration of attack burst	0.4s
Time between attack packets in a burst	0.2s
Mean service time	12s
Interval between legitimate user‘s requests	3s
Number of server threads (modules)	1
Number of positions in service queue	N = 4
Number of attack threads	N = 4
Attack duration (Termination of Simulation)	50,000
RTT	1s

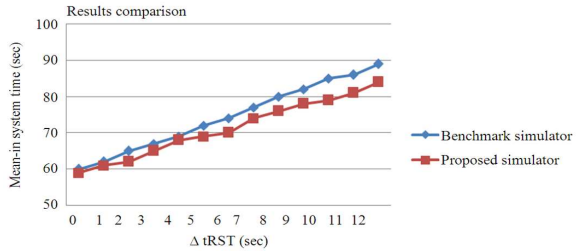

Fig. 6: Comparison of Mean in-system time performance metric in both simulators

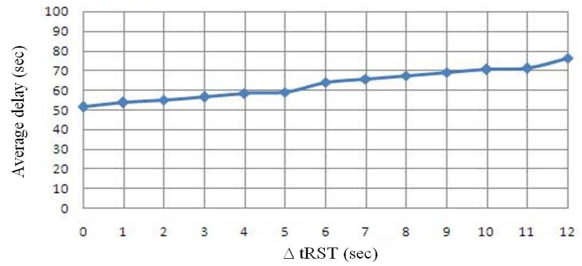

Fig. 7: Average delay in developed discrete event simulator

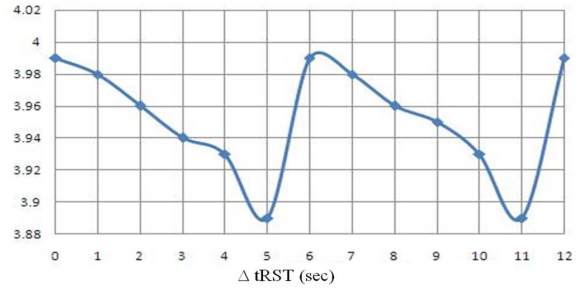

Fig. 8: Average buffer size in developed discrete event simulator

The acquired results are compared with the results obtained from the NS2 based results obtained by (Macia-Fernandez *et al*., 2010). The results for the mean in-system times (Fig. 6) have shown that the value of this performance metric rises by Δt_{RST}. This is due to the fact that Δt_{RST} is added to the service time while the request is being served in the server. The results are consistent and equivalent to the graph gained in (Macia-Fernandez *et al*., 2010) with negligible deviation (Fig. 6). This percentage deviation serves as a validation of the developed simulator. The analysis of the average delay, shows an increase as the Δt_{RST} increases. Figure 7, displays this observations since the rise of Δt_{RST} would result in increasing the time that a request remained in the server. Figure 8 demonstrates that the average buffer size is almost equal to 4 (the maximum queue size) for all experiments. This implies that the service queue is in the overflow state (full of requests) in most cases. Based on our assumptions, most of the time valid and invalid requests (attack requests) compete to seize the positions in queue. So the buffer is kept full during simulation.

CONCLUSION

In this study, a detailed description of the design and development of a newly proposed discrete event simulator for the purpose of providing an efficient and scalable performance analysis tool for the analysis of HTTP low rate DoS attacks has been provided. There are many trade-offs between exploiting general purpose simulation software and utilizing existing simulators. As for the latter, the developer has the opportunity to grasp the fundamentals of discrete event simulation as well as having the space to customize the event characteristics and simulation components precisely and tangibly. In addition, multitudes of decisive factors such as size, memory usage, portability, license, running time, utilizing general-purpose programming languages and ease of use served as major motivation factors which contribute to the constant proliferation of performance analysis tools. The ease-of-use attribute, small-scale, platform-independent (which only needs a general purpose C compiler) and customized C discrete event simulator with remarkably low execution time gracefully confirms this. The future enhancement encompasses the development of other elements of discrete event simulators for varying analysis such as those concerning other security issues such as Intrusion Detection Systems in particular.

REFERENCES

Begg, L., W. Liu, K. Pawlikowski, S. Perera and H. Sirisena, 2006. Survey of simulators of next generation networks for studying service availability and resilience. Technical Report, TR-COSC 05/06. Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand.

Ghazali, K.W.M., and R. Hassan, 2011. Flooding distributed denial of service attacks-A review. J. Comput. Sci., 7: 1218-1223. DOI: 10.3844/jcssp.2011.1218.1223

Guirguis, M., A. Bestavros and I. Matta, 2006. On the impact of low-rate attacks. Proceedings of the IEEE International Conference on Communications, (ICC' 06), IEEE Xplore Press, Istanbul, pp: 2316-2321. DOI: 10.1109/ICC.2006.255115

Jensen, M., N. Gruschka and N. Luttenberger, 2008. The impact of flooding attacks on network-based services. Proceedings of the ARES 2008 3rd International Conference on Availability, Security and Reliability, March 4-7, IEEE Xplore Press, Barcelona, pp: 509-513. DOI: 10.1109/ARES.2008.16

Jin, G., H. Zhang, H. Zhang and Z. Xie, 2009. Simulation research on DoS attacks and defenses mechanisms. Proceedings of the 2nd International Conference on Intelligent Networks and Intelligent Systems, Nov. 1-3, IEEE Xplore Press, Tianjin, pp: 138-141. DOI: 10.1109/ICINIS.2009.44

Kuzmanovic, A. and E.W. Knightly, 2006. Low-rate TCP-targeted denial of service attacks and counter strategies. IEEE/ACM Trans. Netw., 14: 683-696. DOI: 10.1109/TNET.2006.880180

Lin, C.H., J.C. Liu, F.C. Jiang and C.T. Kuo, 2008. An effective priority queue-based scheme to alleviate malicious packet flows from distributed DoS attacks. Proceedings of the 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Aug. 15-17, IEEE Xplore Press, Harbin, 1371-1374. DOI: 10.1109/IIH-MSP.2008.270

Luo, H. and M.L. Shyu, 2005. The protection of QoS for multimedia transmission against denial of service attacks. Proceedings of the 7th IEEE International Symposium on Multimedia, ISM 2005, Dec. 12-14, IEEE Xplore Press, pp: 6-6. DOI: 10.1109/ISM.2005.115

Macia-Fernandez, G., J. E. Diaz-Verdejo and P. Garcia-Teodoro, 2008. Evaluation of a low-rate DoS attack against application servers. Comput. Security, 27: 335-354. DOI: 10.1016/j.cose.2008.07.004

Macia-Fernandez, G., J.E. Diaz-Verdejo and P. García-Teodoro, 2006. Low rate DoS attack to monoprocess servers. Lecture Notes Comput. Sci., 3934: 43-57. DOI: 10.1007/11734666_5

Macia-Fernandez, G., R. A. Rodriguez-Gomez and J. E. Diaz-Verdejo, 2010. Defense techniques for low-rate DoS attacks against application servers. Computer Networks, 54: 2711-2727. DOI: 10.1016/j.comnet.2010.05.002

OMNeT++, 2001-2009. What is OMNeT++? OMNeT++ Community.

OPNET, 2012. OPENT IT Guru network planner network planning and engineering enterprises. OPNET Technologies, Inc.

Reineck, K.M., 2008. Evaluation and comparison of network simulation tools. M.S. Thesis, University of Applied Science Bonn-Rhein-Sieg, Department of Computer Science.

Rhazi, A.E., S. Pierre, and H. Boucheneb, 2007. A secure protocol based on a sedentary agent for mobile agent environments. J. Comput. Sci., 3: 35-42. DOI: 10.3844/jcssp.2007.35.42

Robinson, S., 2004. Simulation: The Practice of Model Development and Use. 1st Edn., Wiley Press, Chichester, ISBN-10: 0470847727 pp: 316.

Shevtekar, A. and N. Ansari, 2006. Do low rate DoS attacks affect QoS sensitive VoIP traffic? Proceeding of the IEEE International Conference on Communications, (ICC' 06), IEEE Xplore Press, Istanbul, pp: 2153-2158. DOI: 10.1109/ICC.2006.255089

SNT, 2008-2012. Simulation, software and services for planning, testing and training. QualNet Network Simulator, Scalable Network Technologies Inc.

SOPHOS, 2011. Security threat report. SOPHOS Ltd.

Viswanathan, A., V.P. Arunachalam and, S. Karthik, 2012. Geographical division traceback for distributed denial of service. J. Comput. Sci., 8: 216-221. DOI: 10.3844/jcssp.2012.216.221

Zhu, Q., Z. Yizhi and X. Chuiyi, 2011. Research and survey of low-rate denial of service attacks. Proceedings of the International Conference on Advanced Communication Technology, Feb. 13-16, IEEE Xplore Press, Seoul, pp: 1195-1198.