Measuring the impact of DDoS attacks on Web Services - A Realtime Experimentation

Sunny Behal*, Krishan Kumar†
*Research Scholar, Punjab Technical University, Punjab, India
†Associate Professor, SBS State Technical Campus, Punjab, India
Corresponding author: S. Behal (email: sunnybehal@sbsstc.ac.in).

***Abstract*—Distributed Denial of Service (DDoS) attack is a severe threat to the timely delivery of extensively used web-based services. Hackers often compromise the vulnerable systems freely available on the Internet for launching DDoS attacks to degrade or sometimes completely disrupt such web-based services causing inconvenience to the end users and massive financial losses to the service providers. To measure the impact of DDoS attacks on these services and its severity, we need to evaluate precisely, quantitatively and comprehensively the DDoS impact metrics. As part of the work, a DDoS Testbed (DDoSTB), which is a hybrid of real and emulated nodes, is developed to generate a variety of attack scenarios using benchmark attack tools and traffic generators. The impact of DDoS attacks on web services is measured in terms of a proposed set of network level, application level, and server level metrics.**

***Index Terms*—Attack Tools;DDoS Attacks ; Experimentation; Network Security;Performance Metrics**

I. Introduction

Over the past decade, advancements in Information and Communication Technology (ICT) have significantly transformed the way in which information is accessed and communicated, particularly via the Internet. Web-based services are used extensively for day-to-day activities. Thousands of companies have deployed Web servers and their usage rates have increased dramatically in recent times. These interactive Internet-based applications must be processed on time to ensure high quality of service. However, these services are delayed, degraded and sometimes completely disrupted because of unavailability of Internet. Even a few minutes of service downtime can lead to not only lost in revenue of these companies but also concerning intangibles such as loss of customer faith, unfavorable media coverage and liability of legal actions. As per the recent report of Kaspersky [1], nearly 40% of all the businesses on the Web experienced at least one DDoS attack, with total financial loss cost in hundreds of millions of dollars during last year. According to CERT [2], "a DDoS attack is a malicious attempt from multiple systems to make computer or network resources unavailable to its intended users, usually by interrupting or suspending services connected to the Internet. Nowadays, almost all the DDoS attacks are launched through Botnets"[3]. DDoS attacks deny the service of a web server by sending a huge amount of useless traffic towards some critical network resource or by exploiting vulnerabilities in the operating system. The attacks of the first type are called flooding DDoS attacks because they exhaust the link bandwidth, reduce the router processing capacity and network resource usage, whereas the attacks of the second type are called vulnerability attacks because they make use of the vulnerabilities of protocols such as TCP and HTTP. They exhaust the server resources like CPU cycles, memory, buffers, etc. Examples of network layer attacks include UDP flood, ICMP flood whereas application layer attacks include HTTP flood, SSL flood, etc. [4], [5].
According to the recent Worldwide Infrastructure Security Report (WISR) [2], the volume of traffic of such attacks has been amplified to around 500 Gbps in the year 2015 as compared to 400 Gbps in the year 2014 and 200 Gbps in 2013. Out of different types of DDoS attacks, the application-layer attacks are increased up to 93 percent in 2015, from 90% last year and 86% in 2013. As per the report, around 70 % of the application layer DDoS attacks targeted customers and web services in 2015 mainly attacking port 80 of web servers.

The key objectives of this paper are as follows:

1) *To propose the taxonomy of network level, application level, and server level performance metrics to measure the impact of DDoS attacks on web services.*
2) *To design and develop a representative DDoS Testbed (DDoSTB) to launch varieties of network layer and application layer DDoS attacks using a hybrid of emulated and real network elements.*
3) *To measure the impact of DDoS attacks on web services using proposed performance metrics.*

This paper is a novel attempt that highlights the importance of real experiments for providing better experimentation alternatives to the irrepressible DDoS problem as compared to the simulation and emulation based experiments. The performance of a web server is investigated in real time under different scenarios of DDoS attacks launched through a set of benchmark DDoS attack tools and traffic generators. The impact of various DDoS attacks is measured with and without attack by evaluating the identified set of the network layer, application layer, and server side metrics.

This paper organization is as follows: Section-II presents related work, section-III briefly summarize the identified performance metrics, section-IV provide the experimental setup, results are analyzed in section-V, and finally, section-VI con-

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

cludes the paper by summarizing the results obtained and highlighting the need for real-time experimentations.

II. Related Work

The timely response of web services to the Internet-based applications is very critical. Most of the researchers have measured the impact of DDoS attacks on web services in simulation and emulation based experiments. However, the validation and testing of these services should be performed in real attack scenarios. DDoS attacks impact metrics are closely related to measuring the effectiveness of the DDoS defense approaches. Therefore, it is vital to measure these metrics in real-time environments. You et. al/ [6] compare application layer throughput i.e. goodput without attack, under attack, and with the defense to measure the impact of HTTP attack. Gupta et al. [7] proposed flow and volume based two statistical metrics to measure impact whereas Jelena et. al.[8] and Sachdeva et. al. [9] measure the impact of network layer attacks using emulation based experiments using DETER testbed. Hakem et. al.[10] proposes ConnectionScore based technique to measure the resistance of a web server against DDoS attacks. Connectionscore is computed in real time, based on the history and statistical analysis. They evaluate the performance of their proposed work on Emulab environment using real traffic traces of ClarkNet WWW server. Hussain et. al [11] develop an evaluation methodology to measure the impact of web services under different categories of attacks using a hybrid of simulation and emulation based experiments. They measure many packet level metrics at client side like server response rate, average response time, and server-error rate whereas, at the server-side, they measure client packet rate, packet-drop-rate and per packet overhead. At the Link level, they measure link and end-to-end throughput, error rates and latencies. In the context of DDoS attacks, they measure attack intensity, attack duration, and goodput. Das et. al [12]propose an HTTP request arrival rate based performance metric to differentiate between random flooding, shrew flooding or flash crowd. Bhatia et. al [13]propose to use system-level metrics such as CPU and memory utilization to measure the impact of DDoS attacks on a web server. Bhandari et. al [14] classified performance metrics into packet level, aggregate level and application level metrics but didn't analyze and measure those metrics by performing any experimentation. As mentioned in [15], the research community lacks the availability of real DDoS datasets. It has made it essential for the researchers to fabricate the attack traffic and generate in-lab datasets synthetically. These facts motivate the authors to develop a representative DDoS Testbed (DDoSTB), which has been designed using a hybrid of real and emulated nodes, real network equipment like routers, switches, etc., with an aim to enhance the experimentation efficacy as compared to experiments using traditional simulation and emulation based techniques alone.

III. PERFORMANCE METRICS

To measure the impact of DDoS attacks, it is important to understand the underlying TCP connection and HTTP requests semantics. HTTP is a generic and stateless TCP/IP based application-level protocol that is used to deliver data (HTML pages, images, videos, etc.) on the WWW. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, how connections establish and how the servers respond to these requests. Any zombie system needs to establish a TCP connection with the web server, which requires a valid IP address. Otherwise, the connection is not established. A web server uses two kinds of connections: HTTP persistent connection and Nonpersistent connection. A persistent connection also called HTTP keep-alive, or HTTP connection reuse, is the idea of using a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair as in the case of HTTP 1.0. The WWW uses HTTP 1.1 persistent connections by default. We have measured the impact of DDoS attacks on the web server by evaluating many metrics at the network level, application level, and server level. The proposed taxonomy of these identified performance metrics is given in Figure-1 and summarized in Table-1. All the performance metrics are evaluated using time series analysis with a time window of 10 seconds. The proposed metrics can be broadly classified into Extrinsic (E) and Intrinsic (I) metrics. Extrinsic metrics are

Performance Metrics		
Network Level	**Application Level**	**Server Level**
No. of Connections (I) • New • Active	Throughput (E) • Goodput • Badput	CPU Load (I)
Round Trip Time (E) • Minimum • Maximum • Average	Transactions (E) • Shortest • Longest • Successful • Failed	CPU Utilization (I) • User CPU • Sys CPU
Avg. Connection Time (E)	Avg. Response Time / Latency (E)	Memory Utilization (I)
No. of Retransmissions (I)	Normal Packet Survival Ratio (E)	
	Active Serve Rate/ Request Rate (E)	

E – Extrinsic I - Intrinsic

Fig. 1: Proposed Taxonomy of Performance Metrics

TABLE I: Summary of Metrics

Sr No	Metric	Description
1	Throughput (β)	$\beta = (L + A)/Tw$, where L represents Legitimate, A represents Attack bits and Tw is the time window used for time series analysis
2	Average Latency(A_L)	$A_L = \sum_{i=1}^{N}(T_{req} - T_{resp}/N$,where T_{req} represents the time at which the transaction begins and T_{resp} represents the time for getting complete response.
3	No. of Active Connections(δ)	No of transactions that are active at any instant of time.
3	Round Trip Time (RTT)	RTTmin, RTTmax, RTTavg
4	Active Serve Rate / Request rate (ρ)	$\rho = (R_{Response}/R_{Requests})$, where $R_{Requests}$ represents no. of GET request generated by the legitimate clients and $R_{Response}$ represents the no of responses (HTTP 200 OK) received by the clients.
5	Percentage Link Utilization (ξ)	ξ represents percentage of bandwidth that is being used for goodput.
6	Failure Rate of Transactions (FRT)	$FRT = (F_t/T_t) * 100$,where Ft represents the failed transaction and Tt represents the total number of transaction.
7	Transaction Duration (T_D)	$T_D = T_{start} - T_{complete}$ where T_{start} is the time when the transaction start and $T_{complete}$ is the time when transaction ends. As part of the work, longest and shortest transactions are also computed per time window Tw.
8	NPSR(η)	$\eta = p_l/(p_l + p_a)$, p_l represents the no. of legitimate packets and p_a represents total no of packets received at victim
9	CPU Load	CPU Load = tr + tw + tb ,where tr is the no. of tasks running, tw is the no. of Tasks Waiting for cpu and tb is the no. of tasks blocked.
10	CPU Utilization (U)	U = B / Tw, where B is Busy time, the amount of time that the server was active during the time window (Tw).
11	Memory Utilization (MEMused)	MEMused = = Total Physical Memory - (Memory Free + Memory Buffers + Cache Memory)

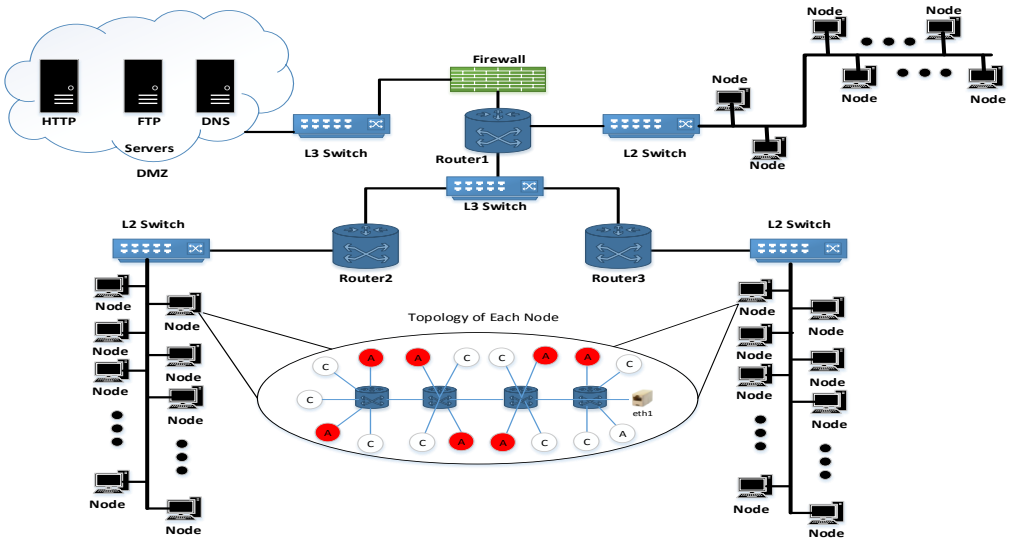


Fig. 2: DDoS Testbed (DDoSTB)

those that can be computed and observed by external entities about the object being measured. Whereas, Intrinsic metrics are those that can be computed by the object itself and by analyzing the internal algorithms and data structures such as queues and connection tables etc.

IV. Experimental Setup

The motivation behind performing real experiments comes from [15], [16] wherein the four validation approaches used for performing experiments such as mathematical models, simulation, emulation and real systems are compared along with their pros and cons. The limitations of existing validation approaches made us focus on real experimentation setup for conducting more meaningful experiments for DDoS defense. In emulation, the real elements like operating system and real applications are combined with simulated elements like soft network links, virtual intermediate nodes, and unrealistic background traffic. Emulation makes use of soft routers for making network connections. Emulation technique has the concern of scalability as it is not feasible to extend the topology of computer systems beyond certain limits and one cannot make network topology as large as of an Internet Service Provider (ISP). On the other hand, real systems provide realistic network conditions, real operating systems, applications and platforms and are proven to be best for network based experimentations. Apart from these advantages, real systems are ideal for validation of any research but they also carry some limitations like (a) change of network topology is not possible for a new experiment, (b) it is very unsafe to do live experiments with Internet worms, viruses, etc. because they can easily escape from the experimentation setup and can damage the live network components, and (c) the flooding based DDoS attacks can cause degradation of network links etc. Real systems are ideal for validating network-based research but they are limited in their functionality because of the difficulty in handling large topologies. We designed a real DDoS Testbed (DDoSTB) using a hybrid of real systems and emulated systems as shown in Figure-2 as one step ahead by mixing more physical network elements as compared to emulation. Our aim is to produce the results as close to real live networks. Emulation is integrated into the experiment to scale the topology, as it is not possible to do experiment with hundreds of physical nodes at one place. This approach eliminates the need for elaborate synchronization of multiple client machines in the experiments. It also provides the complete control over the workload; we can model the homogeneous or heterogeneous clients, and can completely specify their HTTP and TCP behaviors [17]. While designing a real representative testbed, all the dimensions of an actual experimentation setup as mentioned in [11], [8], [18], [19] are kept in mind and followed.

Many benchmark DDoS attacks tools and traffic generators are used as described in [20] for conducting experiments. We follow the principles of benchmarking for designing real experiments [21]. The real D-Link Routers and L2/L3 layer switches are used to construct a backbone network. In our experiment scenarios, we have used httperf[22], [23] and D-ITG traffic generator [24] for the generation of synthetic legitimate traffic and background traffic respectively whereas bonesi[25] attack tool is used to generate UDP flood and Goldeneye attack tool [20] is used to generate application level attack traffic. As part of the work, a representative testbed is designed. We also refer work done in [26], [17] to execute real experiments with 30 physical systems. To increase the number of nodes and requirement for the unique IP's in the network, the capabilities of V-core emulator [27] and IP aliasing [28] are used. In the testbed, we deploy 30 physical systems clubbed in 3 clusters of 10 computers each, which run the Ubuntu and Windows XP instances, three physical D-Link Routers, four L2 switches, two L3 switches and one two-processor 8-

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

TABLE II: Experiment Parameters (Scenario-I)

Experiment Scenario-I	
Experiment Time	300 seconds
Attack Duration	60 seconds
Legitimate Traffic Type	HTTP
Attack Traffic Type	UDP
No. of attack packets	2/10/100 times of Legitimate traffic
Attack Type	Constant Rate
Attack tool	bonesi
Bottleneck Bandwidth	10 Mbps

TABLE III: Experiment Parameters (Scenario-II)

Experiment Scenario-II	
Experiment Time	400 seconds
Attack Duration	60 seconds
Legitimate Traffic Type	HTTP
Attack Traffic Type	HTTP
Attack Type	Impulsive
Attack Tool	Goldeneye
% of Zombies	10/50
No. of Legitimate Clients	100

core web server. One v-core node implements a topology of 30 virtual nodes including eight routers and hence, scaling to overall hierarchical topology (topology equivalent to an ISP) size of 900 nodes with 243 routers in the network. The connection link between the server and router-R3 act as the bottleneck link. The bandwidth of all other links in the topology is 100 Mbps whereas the bottleneck bandwidth is 10 Mbps in the case of scenario-I.

V. RESULTS AND DISCUSSION

We conduct a number of real time attack scenarios using parameters enlisted in Table-I and Table-II.

The results of various scenarios are discussed here.

Scenario-I: A UDP flood is generated using Bonesi attack tool at the varying strengths of 2/10/100 times of legitimate HTTP traffic, with the constant packet size of 40 bytes for 60 seconds. The impact of varying intensity of UDP flood on normal going HTTP traffic is measured using time series analysis with a time window of 10 seconds. The result of this attack is shown in Fig. 3a-3h in the form of various network level and application level performance metrics. The UDP flood starts at 120th second and lasts up to 180th second and rest is the normal traffic. The bottleneck bandwidth is 10 Mbps between the webserver and firewall.

Scenario-II: The high rate network layer DDoS attacks (scenario-I) are easy to detect and mitigate as clear from high packet rates and similarity of packet generation mechanism. To go undetected, the attackers have shifted their focus from traditional high rate network layer DDoS attacks towards more sophisticated legitimate traffic looking application layer attacks. It is tough to detect application layer HTTP attacks because the attackers can mimic the same traffic pattern as that of legitimate users[29]. To investigate such type of DDoS attack, we generate the application layer HTTP attack using sophisticated goldeneye attack tool. The Goldeneye is an HTTP based DDoS attack tool with impulsive behavior. An impulsive attacks is the variation of pulsating attack. It generates a series of random sized packets within specified range and, it is very tough to predict its exact behavior and traffic generation pattern. We generate attack traffic similar to the legitimate-looking HTTP traffic with the packet rate in the range of 7-14 pkts/s. Though there is no known formula of proper mixing of normal and attack traffic [30], we took 10% attackers in the first case and then increase to 50% attackers with a total of 100 legitimate nodes. The HTTP flood starts at 170th second for 60 seconds. The Fig. 5a-5h shows the impact on various network level and application level performance metrics.

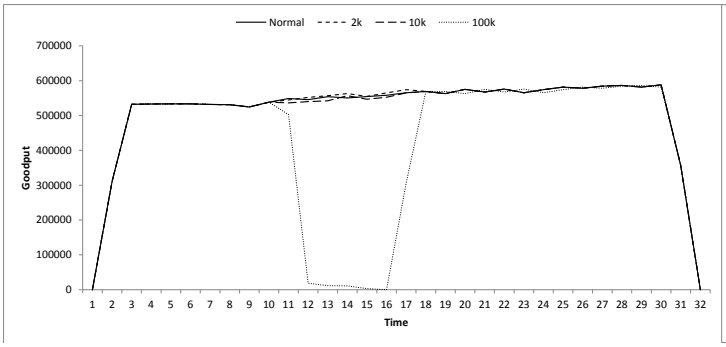
Throughput The throughput is measured in terms of goodput and badput. In the case of low rate network layer flooding DDoS attack (scenario-I), at bottleneck bandwidth of 10 Mbps, the goodput of the system remain almost constant but it plunges to 0 during the high rate UDP attack as shown in Fig. 3a whereas the badput of the system increases with the attack intensity as shown in Fig. 3b. In the case of impulsive DDoS application layer attack (scenario-II), the goodput remains almost constant during normal traffic, but it starts fluctuating, once the attack is launched at 160th sec (Fig. 4a). As the no. of attackers increases from 10% to 50%, the goodput starts decreasing because the legitimate clients get less time of CPU processing. Conversely, badput is almost 0 during the normal traffic, but it climbs up once attack strength increases (Fig. 4b).

Average Response Time: Every application has its quality of service (QoS) needs. For HTTP transactions, it is vital to have a minimum time delay between request and response. According to research in [9] an HTTP transaction is considered to be successful if the operation completes in less than 10 seconds. All other operations taking more time than 10 seconds are considered as failed transactions. The average response time in case of UDP flood attack with bottleneck bandwidth of 10 Mbps is close to 3.2 seconds without attack, but it surges to almost six times when attack strength increases (Fig. 3d). However, in the case of HTTP attack (scenario-3) the Avg. response time also jumps to almost six times as compared with the response time of normal traffic (Fig. 4d).

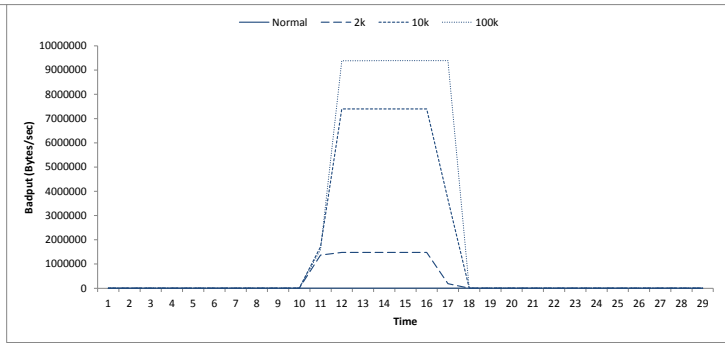
No. of Active and New Connections: Active connections are the number of clients which have completed 3-way handshake (in the case of TCP) and have started sending requests. In the case of bottleneck bandwidth of 10 Mbps, the number of active connections keeps on increasing because all the requests for new connections reached to the web server slowly resulting in piled up of requests at the server queues. Because of choked bandwidth, the active connections start decreasing once attack strength increases. However, if sufficient bandwidth is available, the number of active connections remains equal to new connections. In such case, all the requests from active connections reach at the server in time and process timely. It results in a balance between the number of active connections and new connections. During attack period, the number of active connections increases exponentially because server remains busy in processing useless network layer packets and less time is devoted to the new and active connections, which results in dropping off new connection requests (scenario-II). In the case of HTTP attack, the no. of new and active connections remain

TABLE IV: End to End Delay Statistics

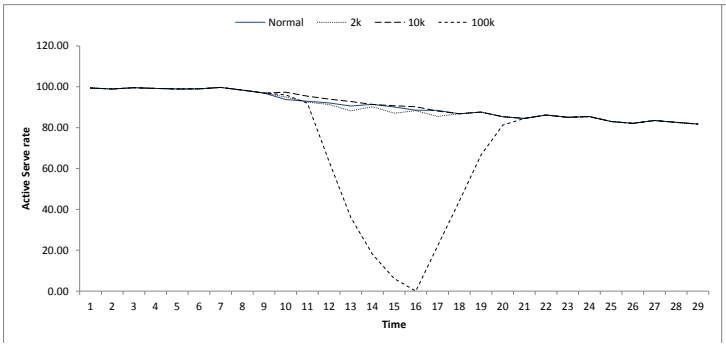
Attack Scenario	RTT		Transaction Duration		Avg. connection Time	
	Min	MAx	Shortest	Longest	Min	MAx
$UDPAttack$	0.0059	2.653	0.013	198.072	0.0029	44.18
$HTTPAttack$	0.0012	0.0285	0.0026	80.42	0.0025	54.41



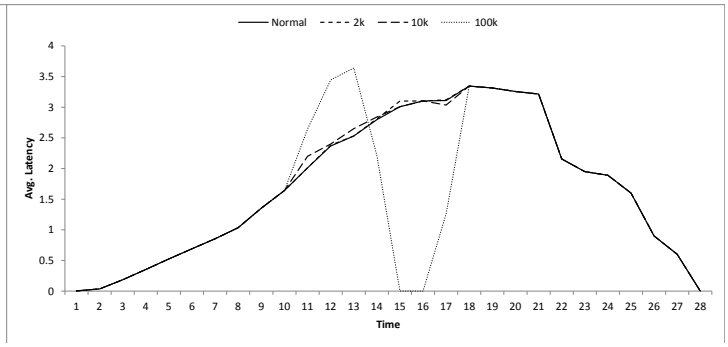
(a) Goodput



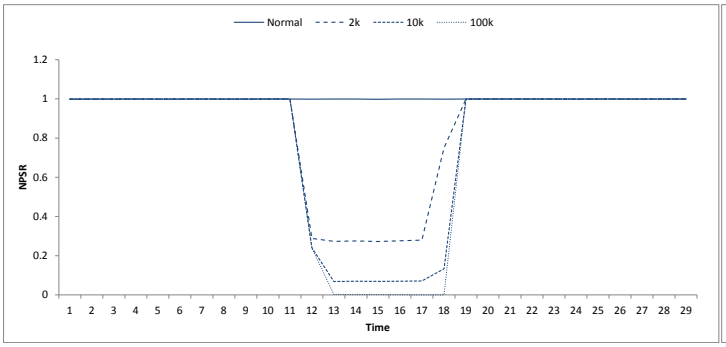
(b) Badput



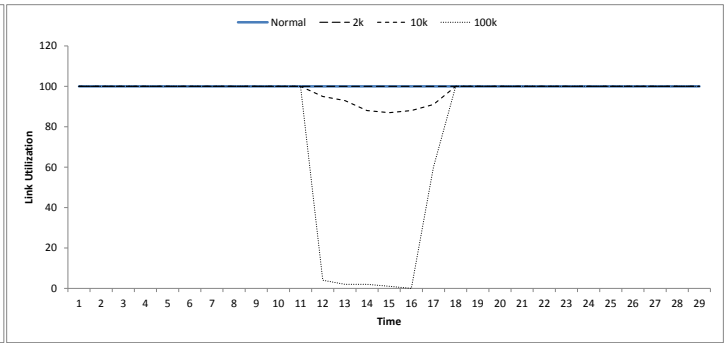
(c) Active Serve rate



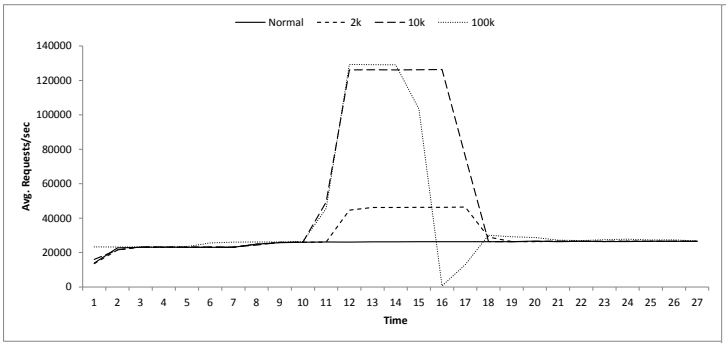
(d) Avg. Response Time



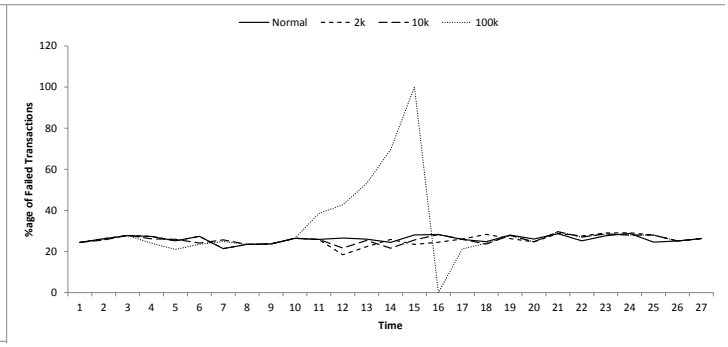
(e) Normal Packet Survival Ratio (NPSR)



(f) Percentage of Link Utilization



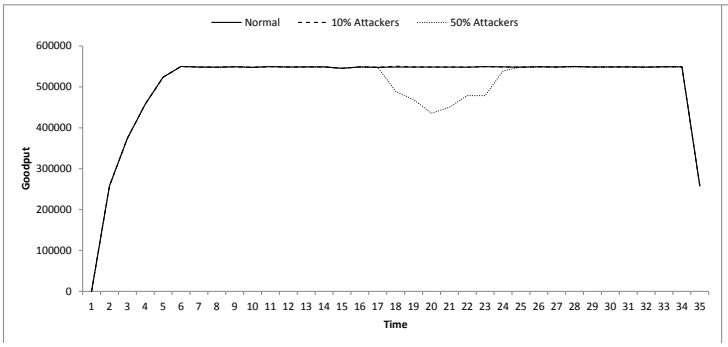
(g) Avg. Request Rate



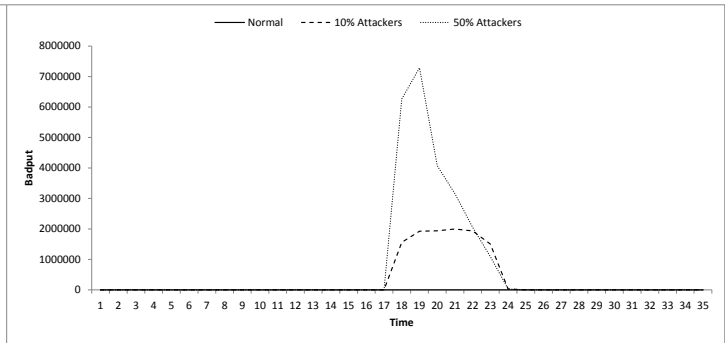
(h) %age of Failed Transactions

Fig. 3: Performance Analysis of Metrics (Scenario-I)

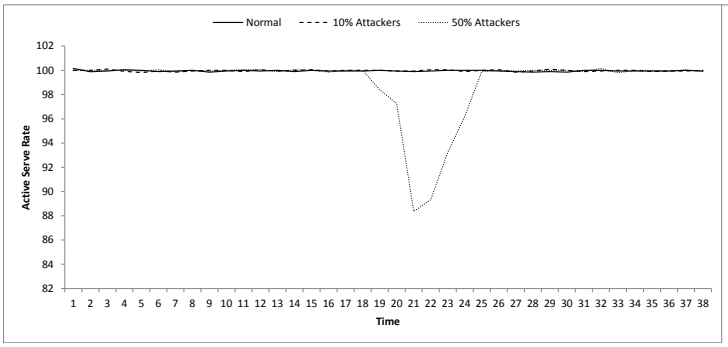
https://sites.google.com/site/ijcsis/
ISSN 1947-5500



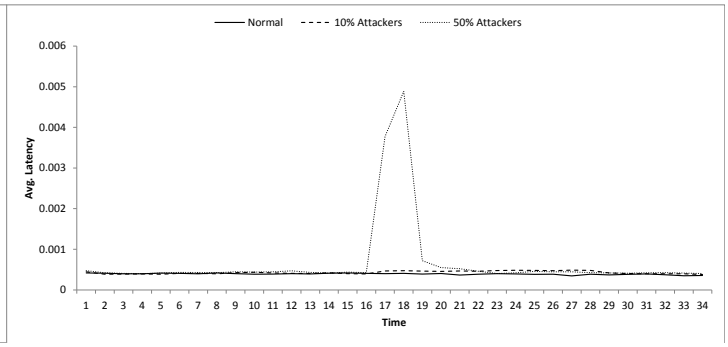
(a) Goodput



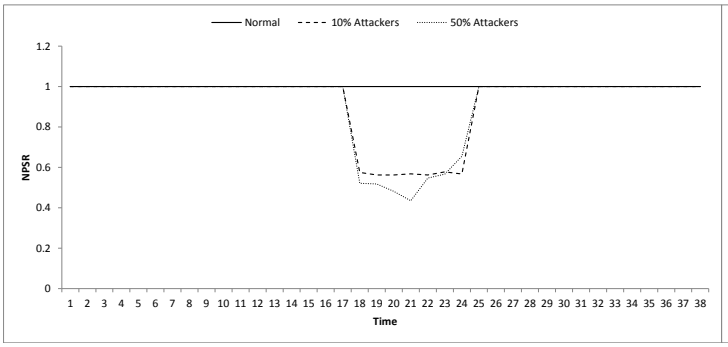
(b) Badput



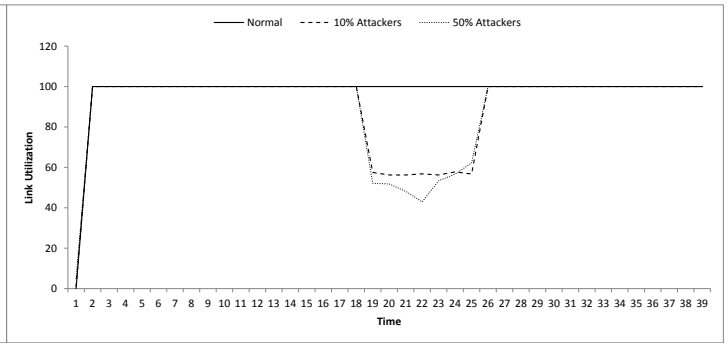
(c) Active Serve rate



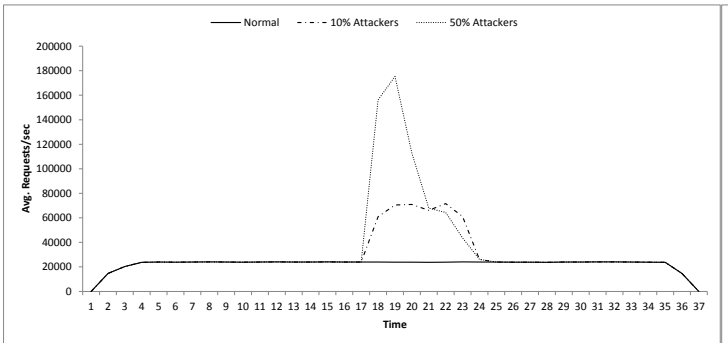
(d) Avg. Response Time



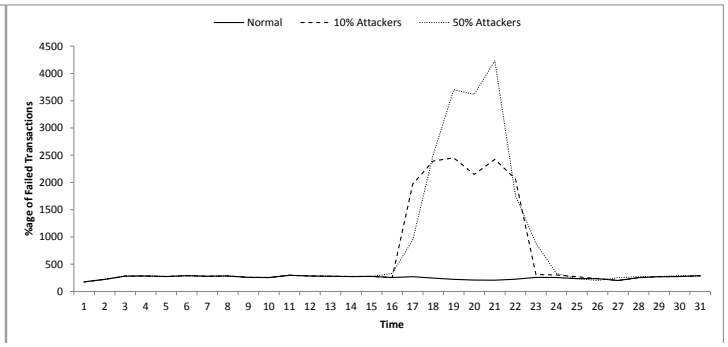
(e) Normal Packet Survival Ratio (NPSR)



(f) Percentage of Link Utilization



(g) Avg. Request Rate



(h) %age of Failed Transactions

Fig. 4: Performance Analysis of Metrics (Scenario-II)

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

almost same in normal traffic but increases with increase in attack strength but they saturate when maximum connection limit of the web server is reached and then starts decreasing if the strength of attack increases further.

Average Connection Time: The average connection time in case of bottleneck bandwidth of 10 Mbps, jumps to more than 20 times of the normal traffic (Table-5), however, in the case of sufficient bandwidth, this parameter remains almost constant. It is observed that the average connection time increases with increase in attack strength.

Failure rate of Transactions There are many reasons for a transaction to fail. Due to congestion at the bottleneck link, lots of legitimate requests can be dropped as shown in Fig. 3h and Fig. 4h. There is an exponential increase in a number of dropped transactions because of timeouts in server queues, resets and dropped connections, as they are not able to reach the web server in time. Hence, the service is denied to the legitimate clients. As we increase the bottleneck bandwidth, the more requests are processed promptly and will be dropped only if buffer overflows at the server's end and maximum connection limit is reached.

Average Serve Rate/Average Request Rate: It is the ratio of average serve rate to average requests generated. Ideally, all the legitimate clients should get the response of their requests send to the web server i.e. the value of this ratio should be equal to 1. But in the case of flooding attack, the web server remains busy in processing bogus packets and legitimate requests are dropped, the value of this parameter decreases considerably (Fig. 3c). In the case of scenario-II (Fig. 4c), as the attack strength increases, less number of legitimate requests reach at the server, causing a decrease in value of this ratio.

Percentage of Link Utilization: Percentage Link utilization is defined as the percentage of bandwidth that is being used for goodput. In the case of bottleneck bandwidth, the link utilization is 100% without attack, however, during high-intensity attacks, this utilization drops to almost 0% (Fig. 3f). The link utilization drops to 50% in case of HTTP attack (Fig. 4f).

Normal Packet Survival Ratio (NPSR): NPSR is the percentage of normal packets that survived the attack. It is the ratio of goodput to the sum of goodput and badput. It is used to measure the impact of attack as a percentage of legitimate packets delivered during the attack. The high value of NPSR ensures that the service continues with no interruption. As shown in Fig. 3e and Fig. 5e), the value of NPSR is 1 when there is no attack but it starts decreases during attack period. As the bandwidth of the link is limited, the legitimate packets start dropping (scenario-I). Even in the case of HTTP attack (scenario-II), the value of NPSR remains 1 during normal traffic but dips to 0.6 when the intensity of the attack is increased.

Server Level Metrics In this section, the impact of different kinds of DDoS attacks on server level metrics is discussed.

CPU Utilization: It is observed that in UDP flood attack, the total utilization of CPU increases with increase in strength of the attack. Whereas in the event of application layer attacks, both the normal and attack traffic is HTTP, the userCPU is utilized more in comparison to sysCPU. Further, with an increase in the attack strength and intensity, more peaks of high CPU utilization are seen (Fig. 5).

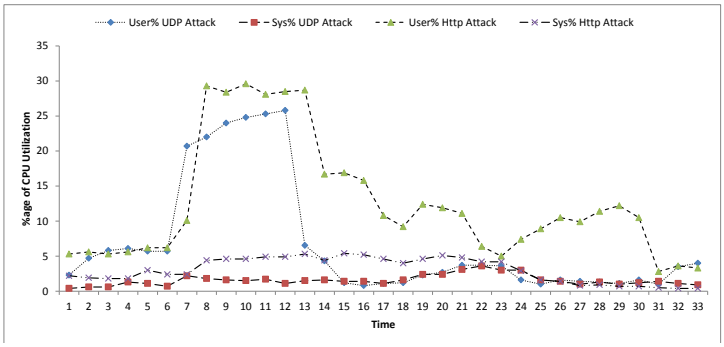


Fig. 5: %age of CPU Utilization

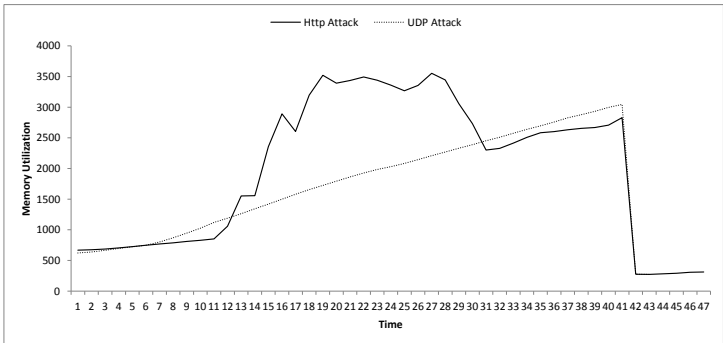


Fig. 6: Memory Utilization

Memory Utilization: It is the %age of primary memory utilized during an attack. As part of their normal execution, applications often allocate and deallocate memory. The available primary memory is 4GB in our case. The value of this parameter remains almost constant during normal traffic conditions (scenario-I). It is observed that the memory consumption increases linearly with increase in attack intensity in the case of UDP attack as compared to the case of HTTP Attack (Fig. 6).

CPU Load: The CPU load is averaged over one minute time interval. The CPU load increases rapidly, as the strength of attack increases (Fig. 7) . In the case of HTTP attack, more CPU load is observed because more new processes and forks are created to handle increased number of requests as

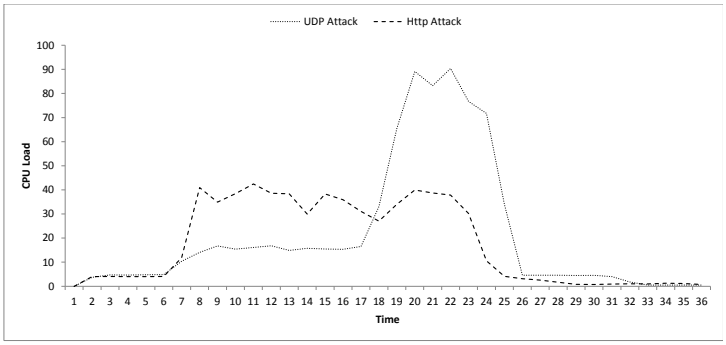


Fig. 7: %age of CPU Load

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

compared to the case in UDP attack.

VI. Conclusion

DDoS attack is a severe threat that leads to unavailability of Internet based web services to the legitimate clients and lead to huge financial losses. As part of the work, a DDoS Testbed (DDoSTB) is developed using a hybrid of real and emulated nodes. The impact of varieties of DDoS attacks on web services is measured using network level, application level, and server level metrics. We generate different attack scenarios at varying attack strengths and bottleneck bandwidths using real benchmark DDoS attack tools. The quantitative measurements clearly indicate the degradation of web services under different varieties of DDoS attacks. Our future work is (1) to propose a detection metric that can characterize the various categories of network traffic, and (2) to scale the topology and capabilities of DDoSTB further, by inculcating more number of virtual and emulated nodes for conducting network experiments at large scale.

Acknowledgments

This Research work has been supported by the All India Council for Technical Education (AICTE), New Delhi, India under the Grant of Research Promotion Scheme(RPS) with Grant No. 8023/RID/RPS-93/2011-12.

References

[1] K. Report, "The kaspersky report 2015," 2015. [Online]. Available: http://business-reporter.co.uk/2015/01/29/firms-face-financial-loss-ddos-attacks-says-report/

[2] A. Networks, "Ddos attack report 2015," 2015. [Online]. Available: http://www.arbornetworks.com/images/documents/WISR2016ENWeb.pdf

[3] G. Gu, *Correlation-based botnet detection in enterprise networks*. ProQuest, 2008.

[4] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004.

[5] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[6] Y. You, "A defense framework for flooding-based ddos attacks," Ph.D. dissertation, 2007.

[7] B. Gupta, M. Misra, and R. C. Joshi, "An isp level solution to combat ddos attacks using combined statistical based approach," *arXiv preprint arXiv:1203.2400*, 2012.

[8] J. Mirkovic, S. Fahmy, P. Reiher, R. Thomas, A. Hussain, S. Schwab, and C. Ko, "Measuring impact of dos attacks," in *Proceedings of the DETER Community Workshop on Cyber Security Experimentation*, 2006.

[9] M. Sachdeva, G. Singh, and K. Kumar, "An emulation based impact analysis of ddos attacks on web services during flash events," in *Computer and Communication Technology (ICCCT), 2011 2nd International Conference on*. IEEE, 2011, pp. 479–484.

[10] H. Beitollahi and G. Deconinck, "Connectionscore: a statistical technique to resist application-layer ddos attacks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 3, pp. 425–442, 2014.

[11] A. Hussain, S. Schwab, R. Thomas, S. Fahmy, and J. Mirkovic, "Ddos experiment methodology," in *Proceedings of the DETER community workshop on cyber security experimentation*, vol. 8, 2006.

[12] D. Das, U. Sharma, and D. Bhattacharyya, "Detection of http flooding attacks in multiple scenarios," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*. ACM, 2011, pp. 517–522.

[13] S. Bhatia, D. Schmidt, and G. Mohay, "Ensemble-based ddos detection and mitigation model," in *Proceedings of the Fifth International Conference on Security of Information and Networks*. ACM, 2012, pp. 79–86.

[14] A. Bhandari, A. Sangal, and K. Kumar, "Performance metrics for defense framework against distributed denial of service attacks," *International Journal on Network Security*, vol. 5, no. 2, p. 38, 2014.

[15] S. Behal and K. Kumar, "Trends in validation of ddos research," *Procedia Computer Science*, vol. 85, pp. 7–15, 2016.

[16] K. Ali, "Algorizmi: A configurable virtual testbed to generate datasets for offline evaluation of intrusion detection systems," 2010.

[17] C. Williamson, R. Simmonds, and M. Arlitt, "A case study of web server benchmarking using parallel wan emulation," *Performance Evaluation*, vol. 49, no. 1, pp. 111–127, 2002.

[18] J. Mirkovic, A. Hussain, S. Fahmy, P. Reiher, and R. K. Thomas, "Accurately measuring denial of service in simulation and testbed experiments," *Dependable and Secure Computing, IEEE Transactions on*, vol. 6, no. 2, pp. 81–95, 2009.

[19] D. Schmidt, S. Suriadi, A. Tickle, A. Clark, G. Mohay, E. Ahmed, and J. Mackie, "A distributed denial of service testbed," in *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*. Springer, 2010, pp. 338–349.

[20] H. Kaur, S. Behal, and K. Kumar, "Characterization and comparison of distributed denial of service attack tools," in *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*. IEEE, 2015, pp. 1139–1145.

[21] J. T. J., "Web server benchmarking," accessed: 20-3-2016. [Online]. Available: http://www.xenoclast.org/doc/benchmark/HTTP-benchmarking-HOWTO/node3.html

[22] D. Mosberger and T. Jin, "httperfxa tool for measuring web server performance," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 3, pp. 31–37, 1998.

[23] J. Midgley, "Autobench: a perl wrapper around httperf for automating benchmarking," *Access: Feb*, 2010.

[24] S. Avallone, S. Guadagno, D. Emma, A. Pescap, and G. Ventre, "D-itg distributed internet traffic generator," in *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*. IEEE, 2004, pp. 316–317.

[25] J. A. Alcorn and C. E. Chow, "A framework for large-scale modeling and simulation of attacks on an openflow network," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–6.

[26] F. Wang, H. Wang, X. Wang, and J. Su, "A new multistage approach to detect subtle ddos attacks," *Mathematical and Computer Modelling*, vol. 55, no. 1, pp. 198–213, 2012.

[27] N. Emulator, "The vcore emulator." [Online]. Available: http://www.nrl.navy.mil/itd/ncs/products/core

[28] H. Pillay, "Setting up ip aliasing on a linux machine mini- howto," 2001.

[29] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 1, pp. 54–65, 2009.

[30] W. Willinger and V. Paxson, "Where mathematics meets the internet," *Notices of the AMS*, vol. 45, no. 8, pp. 961–970, 1998.