Journal of Web Engineering, Vol. 15, No.5&6 (2016) 521-538

DETECTING AND CONTAINING MALICIOUS SERVICES IN AN INTERCLOUD ENVIRONMENT

LOHIT KAPOOR

Thapar University, Patiala, India

Model Institute of Engineering and Technology, Jammu, India

lohit.kapoor@thapar.edu

SEEMA BAWA

Thapar University, Patiala, India

seema@thapar.edu

ANKUR GUPTA

Model Institute of Engineering and Technology, Jammu, India

ankurgupta@mietjammu.in

Received August 18, 2015
Revised April 11, 2016

In the recent past there have been several instances of hackers using cloud computing to launch DoS/DDoS attacks on targets worldwide. The seemingly infinite compute resources on offer make cloud computing an attractive option for launching planetary-scale attacks. Cloud Service Providers (CSP) which rent out computing resources, need to ensure that their platforms are not used by malicious users/services in launching attacks. This paper proposes a novel mechanism for detection and containment of malicious applications based on application profiling. Further, a global blacklist of malicious applications and their performance profiles is maintained and continuously updated to collaboratively aid in quick detection across CSPs. This privacy-preserving scheme effectively neutralizes malicious applications preventing them from misusing the large computational resources on offer.

Key words: Application/service profiling, detecting DoS/DDoS attacks, cloud security
Communicated by: M. Gaedke & B. White

1 Introduction

Recently there has been a spate of security attacks using hired cloud computing infrastructure especially Denial-of-Service (DoS) and Distributed-Denial-of-Service (DDoS) attacks [1, 2]. DoS/DDoS attacks are some of the most widely prevalent attacks on the internet which target well known web-sites/applications/services by overwhelming them with malicious requests affecting their performance or causing them to crash. With cloud computing offering huge amount of computing resources on a pay-per-use basis, it has become easier for malicious users to hire cloud resources and launch DoS/DDoS attacks from multiple locations. This makes it difficult to identify the source of

origin of these attacks and contain their damage [3]. Moreover, some malicious users have used cloud resources to host Attack-as-a-Service which allows third-party users to launch DDoS attacks by just specifying the intended targets [3]. Thus, there is a need to detect malicious applications hosted on the cloud and prevent them from utilizing the vast computing infrastructure of the cloud to launch planetary-scale attacks which can potentially cripple the internet including critical business and government resources.

Most of the work done on detecting DDoS attacks is based on network traffic analysis at the recipient end [4] i.e. at the server hosting the application under attack. It typically involves creating a baseline of normal incoming traffic patterns and detecting any abnormal variations from the known baseline. Once determined that the host is under attack, there is an attempt to drop/ignore packets from the source IP (Internet Protocol) address from where the attack is originating [4,5]. However, performance is definitely degraded and unless high-availability and load-balancing features are implemented to defray the attack, the application/service under attack is severely impacted [6].

This research paper proposes a novel mechanism for effectively and efficiently detecting malicious applications hosted on the cloud. It involves changing the perspective on detecting and containing DoS/DDoS attacks from the recipient to the source. Cloud Service Providers (CSPs) rent out Virtual Machines (VMs) to customers for their computing needs and are responsible if malicious users use their resources to launch global DDoS attacks. Hence, CSPs need to check the malicious use of their resources and the proposed method allows CSPs to detect and contain DDoS attacks originating from their computing infrastructure. In an intercloud environment if each CSP can prevent DoS/DDoS attacks originating from their infrastructure, the entire intercloud becomes free of such attacks. This provides a strong motivation for the CSPs to participate in this collaborative process of detecting and containing malicious services. One conceivable method to detect malicious hosted applications in the cloud is packet sniffing to understand the kind of traffic being generated and transmitted by the hosted application. However, this would constitute a violation of privacy and be unethical on part of the CSP [7]. Hence, a privacy-preserving mechanism is required which is still able to detect whether the hosted application is launching DoS/DDoS attacks using the CSP's compute infrastructure. The proposed mechanism is based on the concept of application/service profiling, which involves creating detailed performance and behavior profiles of hosted code. The run-time behavior of the hosted application is further compared to the performance profiles in a global database of known malicious applications for quick detection. Unknown applications are referred to expert system for classification based on deviations from known normal behavior and the database expanded. Detailed experimental analysis of known malicious applications is used as a basis for initial detection and determining the effectiveness of the proposed scheme. Encouraging results obtained establish the validity of this novel scheme.

The rest of the paper is organized as follows: In Section 2, we present the related work in detecting DoS/DDoS attacks from a cloud perspective. Section 3 presents the system model, followed by a discussion of the experimental setup and results obtained in Section 4. Finally, section 5 concludes the paper and presents some directions for future work.

2 Related Work

A DoS/DDoS attack is an attempt to make the services assigned to the authorized users unavailable. The occurrence of a DoS/DDoS attack typically increases bandwidth consumption, causing congestion,making the service inaccessible to the users or floods the service with packets to overwhelm it. It is thus intuitive to devise mechanisms collocated at the server hosting the service to quickly detect attacks and then attempt to recover from them. For large applications which are hosted in the cloud, the CSPs can provide several layers of security and high-availability features including firewalls, traffic analysis, redundant-hosting, dynamic service-replication and vm migration. However, such attack detection strategies and recovery can be expensive, both in terms of time and resources, while the impact on service performance cannot be entirely mitigated.

Usage of an Intrusion Detection System (IDS) is the most popular method of defense against these types of attacks [8]. It monitors network and/or system activities for malicious activity. The main function of intrusion detection systems is to identify malicious activity, log information about this activity, attempt to block/stop it, and report it. Majority of intrusion prevention systems utilize one of three detection methods: signature-based, statistical anomaly-based, and stateful protocol analysis.The use of IDS to secure services in cloud from DDoS attack is proposed in [9] in which SNORT technique is used. In this method network traffic inbound and outbound are audited. The packets are examined in real-time by the intrusion detection system for a particular type of attack based on predefined rules. A similar approach is used in [10] in which SNORT is loaded onto each virtual machine for sniffing all traffic. This scheme has been shown to perform reasonably well in a Eucalyptus [11] cloud. All IDS-based schemes are designed to work at the recipient end which is the target of the attack. Further, attacks launched by well knownDDoS toolkits like High Orbit Ion Cannon (HOIC) [12], a DDoS http-based attack toolkit, uses what it calls "Booster Scripts" to modify the packet headers and introduce variations in the attacks. By examining the valid header ordering possible abnormalities can be detected. However, inspecting network traffic designated for a hosted application (either at source or recipient) constitutes a violation of privacy in case of normal user traffic and would be unethical on part of the CSP. Hence, a privacy-preserving mechanism is clearly required.

In case of a federated cloud model, a defense federation has been proposed in [13] for guarding against DoS/DDoS attacks. Each cloud is loaded with separate IDS. The different intrusion detection systems work on the basis of information exchange. In case a specific cloud is under attack, the cooperative IDS alerts the whole system. A decision on trustworthiness of a cloud is taken by voting, and the overall system performance is not hampered.

Some CSPs offer DDoS mitigation services [14, 15]to ensure the uptime of the service for customer in the event of DDoS attacks. Prolexic [16] claims to have an effective DDoS mitigation strategy with four DDoS traffic scrubbing centers. All in-bound traffic is routed to the nearest scrubbing center. It also relies on filtering techniques and anti-DoS hardware which closes the source of all the botnet activities. However, routing traffic for scrubbing introduces communication delays for normal traffic possibly impacting response time, SLAs and even user satisfaction.

Authors in [17] use Services Oriented Architecture (SOA) to ascertain the true identity of source of DDoS attack and present a model to filter these packets. However it doesn't explain how to detect source if attacker is using cloud infrastructure for attack.

Thus, most of the work done on detecting DDoS attacks is based on:

- Network traffic analysis at the recipient end i.e. at the server hosting the application under attack.
- IDS Snort rule and attack signatures
- Use of firewalls and puzzle servers.

All the above mechanisms are reactive in nature and entail significant processing costs and communication delays without any performance guarantees for applications under attack. Existing work also does not adequately address preventing the misuse of cloud computing infrastructure for launching DoS/DDoS attacks. The use of cloud computing by attackers affords a unique opportunity for CSPs to detect and contain DoS/DDoS attacks at the source rather than at the destination which has been the focus of existing work. This research paper presents a novel mechanism that flips the perspective of detecting malicious applications at the source rather than the destination, when cloud computing infrastructure is used in launching attacks. It involves creating detailed performance profiles of installed applications/services at the level of individual VM's and matching them against the profiles of known malicious applications. Unknown applications are classified based on observed deviations from known normal behavior and inputs from an expert system which captures classification logic of expert human administrators. Further, the approach is privacy-preserving and proactive enabling faster detection of malicious applications.

3 System Model

At the CSP, an application monitor module is instantiated per VM allocated by the cloud hypervisor to the customer as shown in Fig 1. Its job is to monitor the installed application image (static) and resource usage and behavior (dynamic) of the customer's application running on the VM. The static profile includes the number of files installed, their checksum and size in bytes etc. If the static profile (installed image) of the hosted application matches against the profile any of the known malicious application it is prevented from executing. The dynamic profile of the application includes dynamic resource usage such as CPU, Memory and I/O as a function of time. If this run-time behavior of the hosted application matches against that of any known malicious application (DDoS application) it is terminated preventing the DoS/DDoS attack from being propagated further. Theprofile matching is done through comparison of time-series data (CPU, Memory and I/O) in real-time with the data of known malicious applications.

If the behavior profile of the application does not match any known malicious applications, but the run-time behavior (trend) exhibits anomalous behavior such as an immediate and sustained spike in outbound traffic, hitting a pleateau after some time and rapid decrease in inbound traffic (indicating that the target application is overwhelmed) it is referred to an expert system, which classifies the application as malicious or non-malicious.

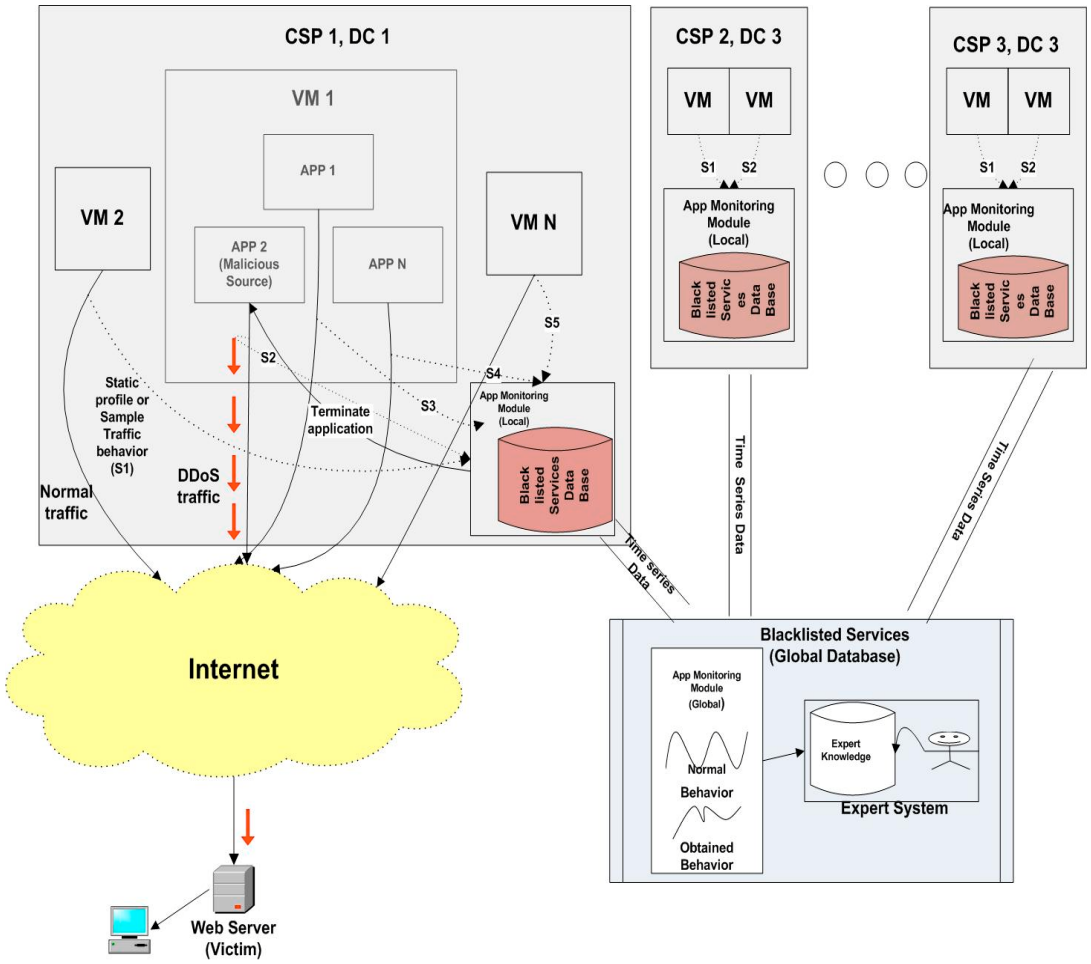

Figure. 1 A Schematic of the proposed mechanism

For well-behaved applications no action is initiated and they are allowed to execute as normal. Following are the sequence of steps involved for profiling and classification of services:

- An "Application Monitor" is instantiated for every hosted application installed on an allotted VM at the CSP. Its job is to create a static profile (installed image) and dynamic profile (run-time resource usage), which is logged in a local log file and compared in real-time to the behavior profiles of known malicious applications stored in a local database. However the global database of known malicious applications is designed to create collaborative intelligence across CSPs on the behavior profiles of known malicious applications aiding in quick detection.

- If the static profile of the application matches, the application is prevented from execution, else it is allowed to execute.
- The run-time resource usage in terms of Memory and I/O usage (inbound and outbound) is stored as a 3-tuple in a time-series database such as rrdtool [18] to optimize data storage and retrieval. The data which is logged locally is also simultaneously compared with the global database storing time-series data for known malicious applications. The global database is initially seeded with the time-series data obtained from experimental analysis of available and known malicious applications and grows further as new malicious applications are detected and classified.
- If no matches are found, but network traffic generated exhibits anomalous behavior, the application is referred to an expert system to help classify the application as malicious or normal and the global database of malicious applications is updated for future reference if application is found to malicious.

4 Experimental Setup and Results

We hired a total of nine virtual machines with different configurations i.e. three VMs each from Windows Azure [19], Amazon EC2 [20] and GoGrid [21] respectively. Further, we created our own Local Cloud (LC) using OpenStack [22] with Red Hat Enterprise and deployed three virtual machines with different configurations. Configurations of VMs used in the experiments are shown in Table 1.

Table 1: Configuration of VMs

CSP Name	Type of VM	Number of cores vCPU	Memory (GB)
Microsoft Azure (CSP1)	A series-A1 (VM1)	1	1.75
	A series-A2(VM2)	2	3.5
	A series-A2(VM3)	4	7
Amazon EC2 (CSP2)	m3.medium (VM1)	1	3.75
	m3.large (VM2)	2	7.5
	m3.xlarge (VM3)	4	15
GoGrid (CSP3)	small (VM1)	1	1
	medium (VM2)	2	2
	large (VM3)	4	4
Local Cloud (CSP4)	small (VM1)	1	2
	medium (VM2)	2	2
	large (VM3)	4	2

This was done to evaluate dynamic application behavior in different environments i.e. hardware configurations, platforms and under varying conditions. We considered ten commonly available DoS/DDoS attack launching tools, as shown in Table 2, on each of the configured VM instances. These DoS/DDoS tools are capable of generating UDP, TCP SYN and HTTP flooding attacks. After installing these tools we first generated and store their MD5 hashes (checksum) by using "MD5 and SHA Checksum Utility" tool [23]. After this each of these DoS/DDoS tools are continuously monitored on parameters like CPU utilization, Memory consumption, Inbound Traffic and Outbound Traffic by perl scripts built on top of standard tools i.e.rrdtool [18] which is a database tool to work with time-series data.

Table 2: Tools and Attack Type

DDoS Attack Launching Tools	Attack Type
HOIC [12]	http
LOIC [24]	(TCP, UDP, http)
XOIC [25]	TCP, UDP
Hulk [26]	
R-U-Dead-Yet[27]	HTTP post
Tor's Hammer[28]	Http post
PyLoris[29]	HTTP, FTP
OWASP DOS HTTP POST[30]	TCP
DAVOSET[31]	TCP, http
GoldenEye HTTP Denial Of Service Tool[32]	TCP, http

After deploying and executing these DDoS tools we observed and recorded their behavior patterns. For illustration purposes we present the detailed analysis of HOIC [12] and LOIC (Low Orbit Ion Cannon) [24]. The dynamic resource usage of HOIC for memory, CPU, outbound and inbound traffic is shown in figures 2, 3, 4 and 5 respectively. We observe that the memory consumption trend in Figure 2 remains linear with average increase of 0.8 MB per second for different VMs with an observed variation of ±2%. This is intuitive as any application executes same set of instructions and its memory growth can be expected to remain same if executed on similar architecture platforms. In the case of CPU utilization as shown in Figure 3, the trends remain the same but the percentage variation across different VMs is higher due to varying number of cores i.e. for 4 cores the average CPU utilization across CSPs is 6%, for 2 cores the utilization is around 12% and for 1 core the average CPU utilization is 17%. Due to this reason CPU usage is not a definitive parameter to be used for dynamic profile matching. HOIC consumes very high bandwidth as shown in Figures 4 and 5. This is again expected since DoS/DDoS attacks rely on overwhelming their targets. It is observed that in the outbound traffic for first three seconds the average traffic grows very high from 1200 B/s to 10000B/s

and after that it comes down at the rate of 1000 B/s. The inbound traffic, during first three seconds grows exponentially i.e from 1000000 B/s to 1900000 B/s. This is because the inbound traffic depends upon the number of participating users at that time. These trends are repeated across multiple experiments and hence bandwidth (inbound and outbound) emerges an important parameter in detecting malicious applications.

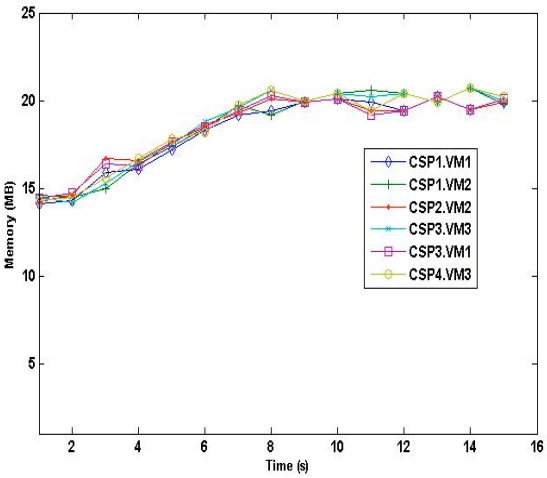

Figure 2: HOIC Memory Consumption

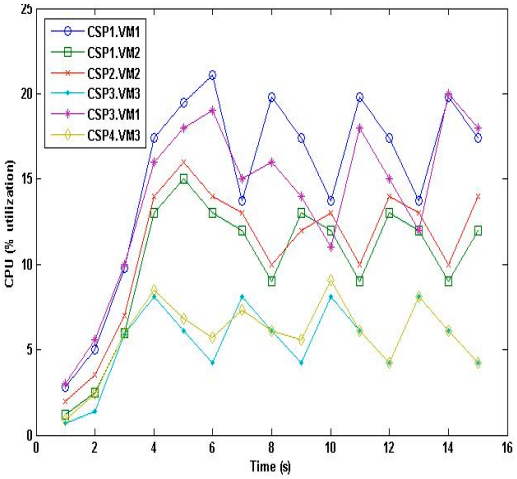

Figure 3: HOIC CPU Utilization

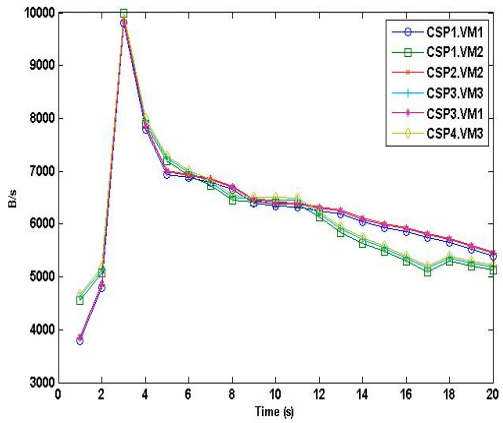

Figure 4: HOIC outbound traffic

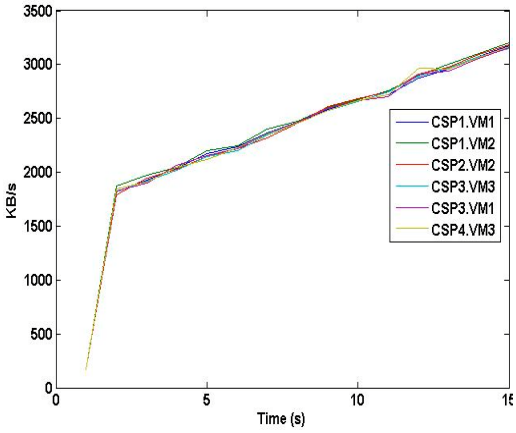

Figure 5: HOIC inbound traffic

The dynamic resource usage profile of LOIC [24] (a TCP/UDP based DDoS attack tool) is presented in Figures 6 through 9. Its memory consumption trend is similar to HOIC with less consumption of memory i.e. on average 10 MB. Further its CPU utilization varies significantly i.e. for any CSP's VM3 (4 cores) during first 7 seconds the average CPU utilization is 1.5% and it grows significantly to 15% thereafter. Further for each VM2 of any CSP (2 cores) the average CPU utilization is 3% initially, increases rapidly to 30% and then remains stable. For VM1 belonging to any CSP the average CPU utilization is 5% and it grows up to 63%.

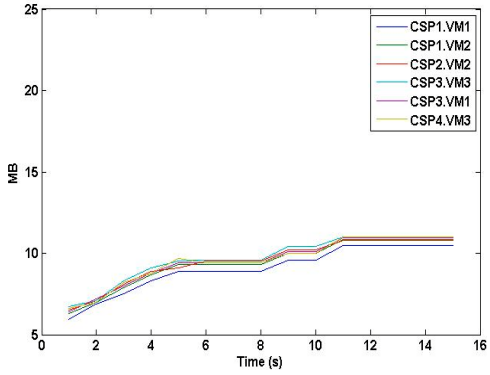

Figure 6: LOIC Memory

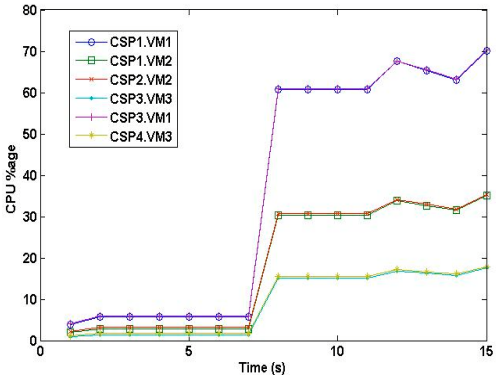

Figure 7: LOIC CPU Utilization

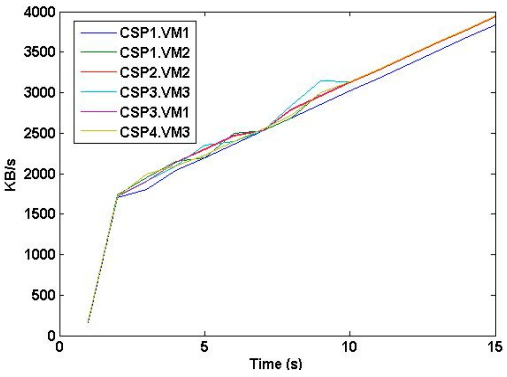

Figure 8: LOIC outbound

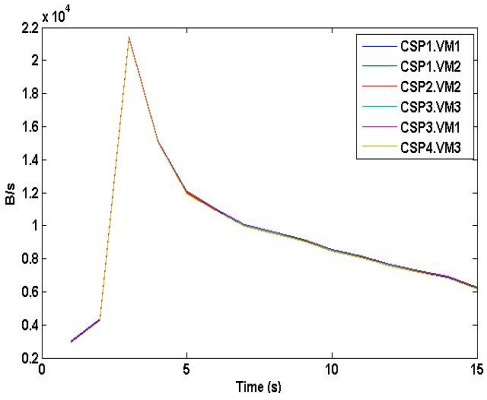

Figure 9: LOIC inbound

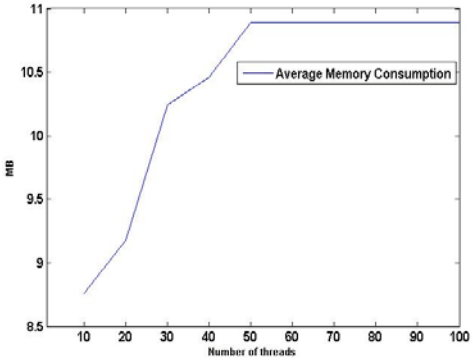

Figure 10: LOIC Memory consumption with variable threads

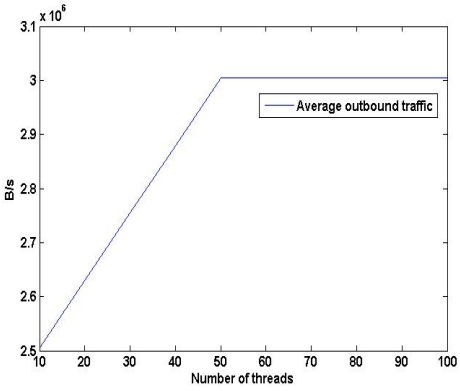

Figure 11: LOIC outbound traffic with variable threads

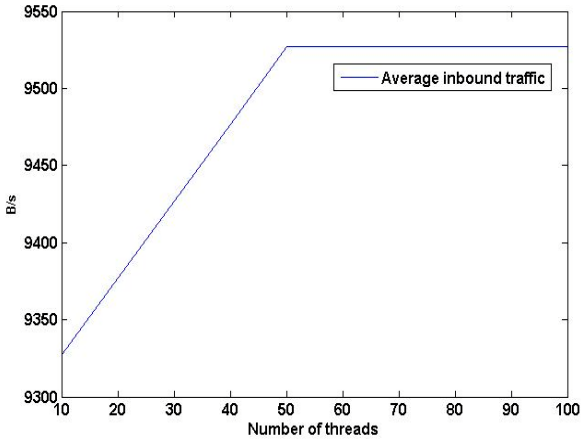

Figure 12: LOIC inbound traffic with variable threads

In Figure 8 it has been observed that LOIC's outbound traffic is very high but predictable i.e. during first 2 seconds it reaches 500000 B/s, then in next 1 sec it shoots to 1900000 B/s with an increase of 100000 B/s for every consecutive second. In the case of inbound traffic it is observed that during first three seconds it is very high and reaches 22000 B/s. This is due to heavy requests sent to the victims' end and the responses received from the victim. After three seconds the response decreases linearly at 500 B/s due to heavy incoming requests until it becomes unresponsive and the DDoS attack succeeds.

We also measured the impact of number of threads on resource usage (Figures 10 through 12) and found that if we double the number of threads, an average increase of 5% in consumption of memory and B/W is observed. After 50 threads on average resource consumption on a VM flattens out. However, the trends of resource usage remain the same and serve as a strong basis for detecting malicious services.

On the basis of the observed behavior, the following strategy emerges:

a) The trends of memory consumption and bandwidth usage (inbound or outbound) can be used as definitive and reliable parameters to identify a malicious application/service.CPU utilization varies significantly across CSPs and hence cannot be used reliably.
b) Bandwidth usage (inbound or outbound) of every http, TCP/UDP based DoS/DDoS tool is very high and it grows exponentially during first five seconds hitting a long plateau later. Hence, bandwidth analysis is an effective indication of identifying malicious behavior.
c) After executing a malicious application, a point comes when the resource usage stabilizes and flattens out after hitting a peak (this usually indicates that the target application has been overwhelmed). We term this as the Point of Confidence (PoC), where it can be safely concluded that the application is indeed malicious. For normal applications the resource usage varies and never hits a long plateau.

d) We store the time series data using rrdtool [18] as a 3-tuple (Memory, Inbound and Outbound traffic) representing observed values of these parameters by the Application Monitor module in the last one second interval. Each data point is stored by rrdtoolas a double float (8 bytes) and hence each 3-tuple occupies 24 bytes of storage.

e) A sampling time of 1 second is determined for logging of application resource usage as time-series data. The 1 second window allows a meaningful trend to emerge and also provides enough time to log and compare data using rrdtool, which typically takes around 2 ms (for retrieval and comparison).In [40] authors observe an average latency of133ms between two geographical locations, 20000km apart which is an extreme case. Even with this extreme latency factored in, a data matching operation against the global malicious database is expected to take an average time much lower than the 1 second window for comparing time-series data.

f) We define "θ", the error interval as ±2% to account for variations in resource usage trends across different CSP platforms.

g) We use the PoC as a reference point to determine the length "L" up to which values are to be stored in time series databases. This optimizes the data storage requirements further and speeds up the detection process. In our experiments a value of 5 for L was the minimum required to correctly match two time-series trends. Greater values of L would enhance the accuracy, but the delay might lead to the target application being irrevocably impacted.

h) Similarity between two time series can be obtained by two methods: i) by measuring and comparing *Euclidean distance* [33]. For example in two dimensions the Euclidean distance is computed as:

$$\sqrt{{}^{m}\Sigma_{i=1}((T_{i,x} - T_{i,x})^2 + (T_{i,y} - T_{i,y})^2)}$$

Where $T_{i,x}$is the observed data point at time x and $T_{i,y}$the observed data point at time y

and ii) by comparing *absolute values* of both time series at the same timestamp. The observed average computational time to calculate and compare Euclidean sum is '2ms' and absolute value is 1ms [34]. Based on the experimental data obtained, we utilize the Euclidean Distance approach for comparing trends of bandwidth usage, while we use the absolute value comparison for comparing memory.

We analyzed the behavior patterns of normal high bandwidth consuming applications like online gaming clients, HD TV, video (720p, 360p, 210p and 120p) uploading/downloading etc. We observed that despite of high B/W consumption per second, very stable and consistent behavior for both inbound and out bound traffic is observed. Results for online game Crazy Taxi [35] are shown in Figures 13 and 14 with an average inbound B/W usage of 64000B/s and outbound usage of 89600B/s.

Therefore, the traffic behavior pattern of DDoS is discernible against normal applications. This is linear, since the percentage increase is only 34 % per second in the case of inbound traffic and 42% in the case of outbound traffic which is very less as compared to observed DDoS tools traffic behavior i.e. on average rise 400% per second. There is an increase in first 5 seconds as well, but this

increase is very low as compared to HOIC or LOIC. However after 6 seconds the traffic remains stable while in the case of DDoS tools its keeps on increasing with high volume.

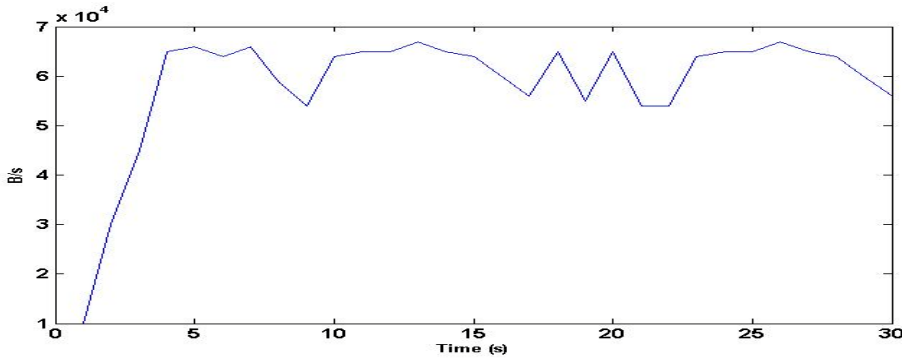

Figure 13: online game (Crazy Taxi) inbound traffic

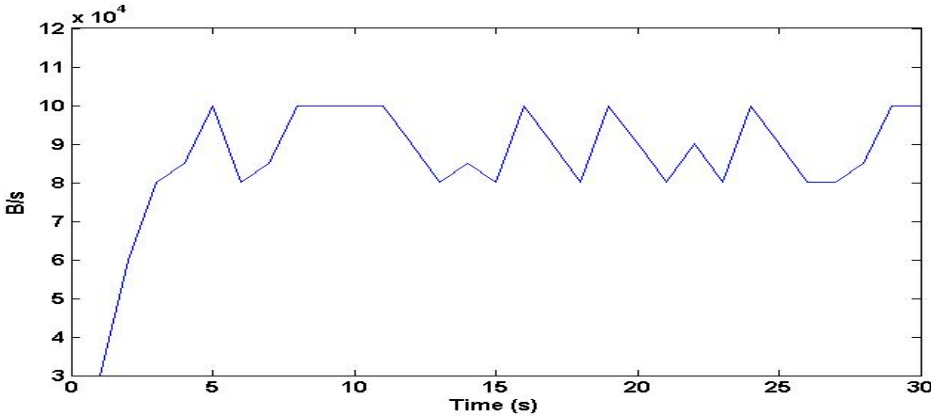

Figure 14: online game (Crazy Taxi) outbound traffic

4.1 Experimental Evaluation

To properly evaluate the effectiveness and efficiency of the proposed scheme we performed an experiment in which ten malicious DDoS tools and 20 different genuine applications were executed randomly on our local cloud setup as described in Table 1. Success of the proposed scheme is defined as the ability to correctly identify malicious applications, while false positives and negatives are recorded to evaluate its effectiveness.

Detection of false positive is given by

FP = Total number of observed false positives/total number of services executed (T),

Similarly detection of false negatives is given by

FN = Total number of observed false negatives/total number of services executed (T),

Also success ratio is given by

$$S_R = 1 - (FP + FN)$$

We define θ=±2 and L=5 for all experiments, the comparison of time-series data of the executing application with the global database of known malicious applications is an iterative process with L iterations. Each successive iteration focuses on a narrower set of matching profiles, disregarding applications where the trends of bandwidth usage and absolute values of memory usage do not match; Table 3 depicts the decision matrix depending upon the outcomes of the comparison process between the running application and the global database. If no match is found but service is consuming high bandwidth it is referred to the classification module which may entail human intervention.

Table 3: Decision Matrix

Memory-usage matches	**Outbound traffic trend matches**	**Inbound traffic trend matches**	**Traffic higher than normal**	
Yes	Yes	Yes	NA	Kill Application
No	Yes	Yes	NA	Kill Application
No	No	No	Yes	Classification Logic + Expert System
Yes	No	No	Yes	Classification Logic + Expert System
Yes	No	Yes	NA	Kill Process
Yes	Yes	No	NA	Kill Process
Yes	No	Yes	NA	Kill Process

The decision matrix above is implemented for a total of 30 deployed applications, including 10 malicious applications. Figures 15 through 21 present the results (success ratio, false positives and negatives) of using a combination of different parameters (m) on the success ratios achieved. The best results are obtained with m = 3 i.e. memory usage, inbound traffic and outbound traffic with a success ratio of 100% achieved, while with m=2 success ratio is on average 90%. This concludes that with the increase in the number of reliable parameters the effectiveness of the scheme increases.

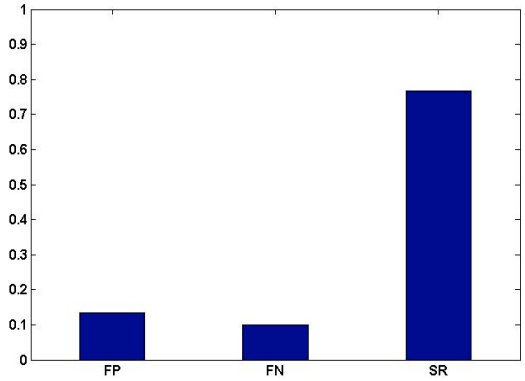

Figure 15: Results with m=1 (outbound traffic)

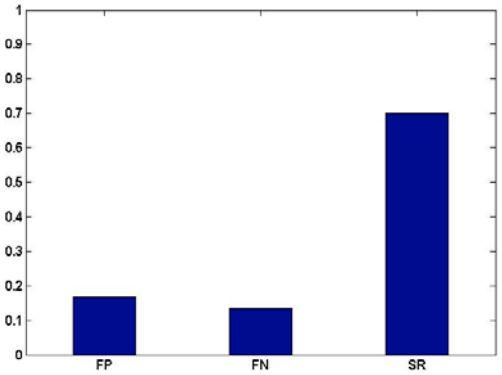

Figure 16: Results with m=1 (inbound traffic)

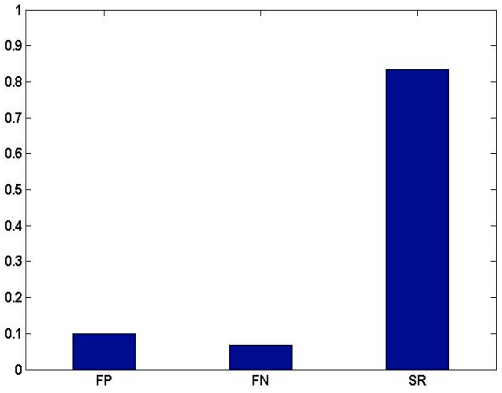

Figure 17: Results with m=1 (memory)

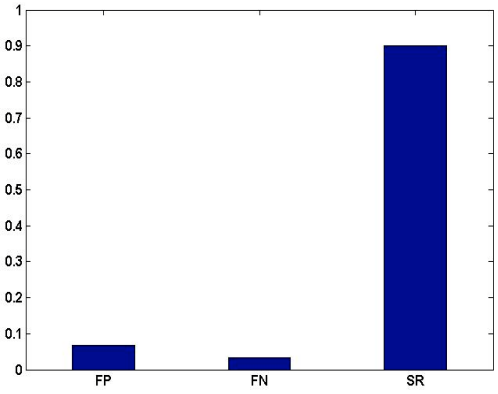

Figure 18: Results with m=2 (memory, inbound traffic)

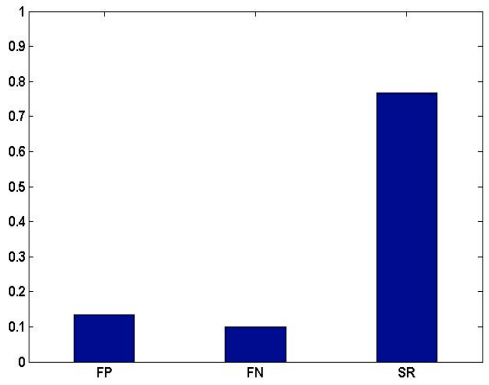

Figure 19: Results with m=2 (memory and outbound traffic)

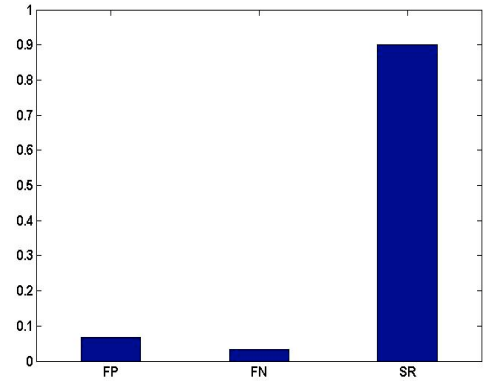

Figure 20: Results with m=2 (inbound and outbound traffic)

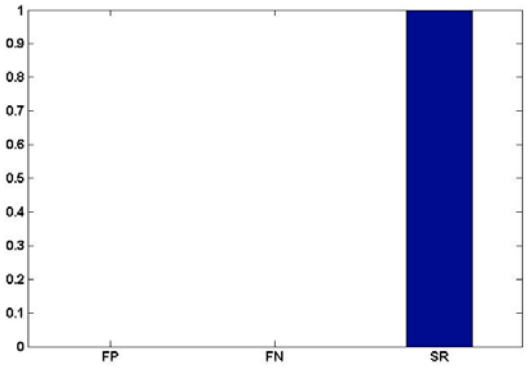

Figure 21: Results with m=3 (memory, inbound and outbound traffic)

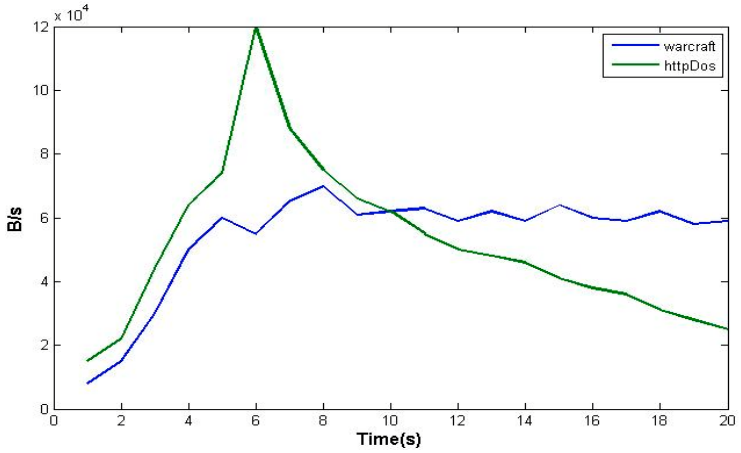

Figure 22: Inbound Traffic trends for warcraft (online game) and httpDos (malicious tool)

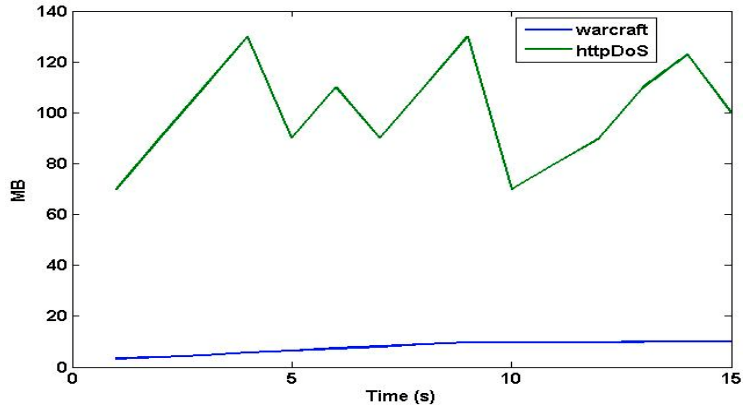

Figure 23: Memory Consumption trends for warcraft (online game) and httpDos (malicious tool)

Figure 22 and 23, depicts the trends of traffic for an online game application named warcraft [36] which incidentally is detected as a False Positive case by the proposed mechanism (for the case m=2 above) because its observed behavior trend is similar to httpDoS tool (a low intensity http based DoS tool). This is due to the reason that trend remains same for the first five seconds. However if we increase the point of confidence to 8 sec (L = 8) the proposed scheme will correctly evaluate its non-malicious behavior. But waiting for a longer duration may affect the target application irrevocably. Hence, the choice of L represents a tradeoff between early detection and accurate detection. However, when memory usage is also considered (m=3 case above) it is observed that memory consumption of these two applications is completely different i.e. for warcraft its 100 MB usage on average while for httpDoS it's 6MB. Therefore, the proposed scheme successfully differentiates normal applications from malicious ones.

The cost of finding the matches in rrdtool is O(P + 2m), where P is the total number of comparison operations between the elements of Time series 'R' and 'S' with m as the length of R and S. The cost to insert each element of R into the time series database is O(m). While this cost is $O(m^2)$ in the *worst case*, in the general case, P will be much less than 2m for common values.

The storage requirement for the time-series data for ten malicious applications in rrdtool for our experimental setup is 1200 Bytes with five data points per application, each occupying 24 bytes. As per [37, 38, 39] there are around 30 types of known DDoS attacks many of which can be launched using tools discussed in [39]. These tools exhibit similar architecture with minor modifications; hence the behavior or attack pattern remains largely similar. Thus, global storage requirements for a couple of hundred identified malicious applications which are capable of launching planetary scale DDoS attacks utilizing cloud infrastructure are not envisaged to exceed a few megabytes.

6 Conclusion and Future Work

This paper proposes a novel mechanism to prevent misuse of cloud infrastructure for launching DoS/DDoS attacks. It flips the traditional perspective by detecting and neutralizing DoS/DDoS applications at the source i.e. CSP rather than at the victim's end. We evaluated and observed the behavior of existing DDoS launching tools to create detailed performance profiles of such applications. Encouraging results have been obtained by detecting malicious applications through comparison of time-series data pertaining to memory usage, inbound/outbound traffic of executing applications. Another important aspect of this mechanism is its privacy-preserving operation as it does not sniff or inspect packets like traditional methods.

As the number of VMs increase, the number of queries to match time-series data against the database of malicious applications is expected to grow correspondingly. It would therefore be required to dynamically scale the rrdtool database instances accordingly with synchronization (between rddtool instances) and load-balancing (for directing queries to the nearest instance) mechanisms in place. Future work shall also involve strengthening the expert system for classification so as to reduce human intervention. We believe that effective collaboration and information sharing about the behaviour patterns of malicious content can prevent the misuse of cloud infrastructure in a big way. Finally, an optimization that we are considering in future is to monitor only applications from non-trusted sources

thereby reducing the query load generated within the system. However, a trust-management mechanism will have to be devised for that which is again beyond the scope of the current work.

References

1. http://wraltechwire.com/report-chinese-hackers-using-cloud-to-spy-on-u-s-/13134654/ (13-Feb-2015)
2. http://www.businessweek.com/news/2013-11-20/chinese-hackers-seen-exploiting-cloud-computing-to-spy-on-u-dot-s (13-Feb-2015)
3. Sabahi, F., "Cloud computing security threats and responses", International conference onCommunication Software and Networks (ICCSN), IEEE,2011, pp: 245-249
4. ChiragModi, Dhiren Patel, Hiren Patel, BhaveshBorisaniya, Avi Patel, MuttukrishnanRajarajan, "A survey of intrusion detection techniques in Cloud", Journal of Network and Computer Applications, Elsevier, 2013, pp: 42-57.
5. S. Subashini , V. Kavitha, "A survey on security issues in service delivery models of cloud computing", Journal of Network and Computer Applications, Elsevier, 2011, pp:1-11.
6. http://cloudtimes.org/2013/06/22/attack-as-a-service-criminals-in-the-cloud/ (13-Feb-2015)
7. Lachlan James, Alice Hutchings and Russell G Smith, "Cloud Computing Threat for Small Business-Final Report", Australian Research Council, Center of Excellence in Policing and Security and Australian Institute of Criminology, 2012, https://www.academia.edu/3620146/Final_report_Cloud_computing_threat_assessment_for_small_business
8. Bakshi A, Dujodwala YB, "Securing cloud from ddos attacks using intrusion detection systemin virtual machine". In: Proceedings of the 2010 second international conference on communication software and networks, ICCSN'10, IEEE, 2010, pp 260–264.
9. AmanBakshi, Yogesh B. Dujodwala, "Securing cloud from DDoS Attacks using Intrusion Detection System in Virtual Machine", ICCSN '10 Proceeding of the 2010 Second International Conference on Communication Software and networks, pp. 260-264, 2010, IEEE Computer Society, USA, 2010. ISBN: 978-0-7695-3961-4.
10. Claudio Mazzariello, Roberto Bifulco and Roberto Canonico, "Integrating a Network IDS into an Open Source Cloud Computing Environment", Sixth International Conference on Information Assurance and Security, USA, pp. 265-270, Aug. 23-25, 2010. DOI: 10.1109/ISIAS.2010.5604069.
11. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S.Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system", in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09), pp. 124–131, 2009.
12. http://sourceforge.net/projects/highorbitioncannon/
13. Claudio Mazzariello, Roberto Bifulco and Roberto Canonico, "Integrating a Network IDS into an Open Source Cloud Computing Environment", Sixth International Conference on Information Assurance and Security, USA, pp. 265-270, Aug. 23-25, 2010. DOI: 10.1109/ISIAS.2010.5604069.
14. http://www.rackspace.com/managed_hosting/services/security/ddosmitigation/(13-Feb-2015)
15. http://www.cloudflare.com/ddos(13-Feb-2015)
16. http://www.prolexic.com/why-prolexic-best-dos-and-ddos-scrubbing-centers.html(13-Feb-2015)
17. LanjuanYang,Tao Zhang, Jinyu Song, JinShuangWang, Ping Chen, "Defense of DDoS Attack for Cloud Computing", IEEE international conference on Computer Science and Automation Engineering, 2012, pp: 626-629.
18. http://oss.oetiker.ch/rrdtool/
19. http://azure.microsoft.com/en-in/
20. http://aws.amazon.com/ec2/

21. http://www.gogrid.com
22. https://www.openstack.org/
23. https://raylin.wordpress.com/downloads/md5-sha-1-checksum-utility/
24. http://sourceforge.net/projects/loic/
25. http://sourceforge.net/projects/xoic/
26. http://packetstormsecurity.com/files/112856/HULK-Http-Unbearable-Load-King.html
27. https://code.google.com/p/r-u-dead-yet/
28. http://packetstormsecurity.com/files/98831/
29. http://sourceforge.net/projects/pyloris/
30. https://code.google.com/p/owasp-dos-http-post/
31. http://packetstormsecurity.com/files/123084/DAVOSET-1.1.3.html
32. http://packetstormsecurity.com/files/120966/GoldenEye-HTTP-Denial-Of-Service-Tool.html
33. http://www.cut-the-knot.org/do_you_know/far_near.shtml#euclidean
34. Luca Deri, Simone Mainardi1, and Francesco Fusco, "TSDB: A Compressed Database for Time Series", TMA 2012, LNCS 7189, 2012, pp. 143–156
35. http://crazy-taxi.en.softonic.com/
36. http://world-of-warcraft.en.softonic.com
37. https://www.rivalhost.com/blog/12-types-of-ddos-attacks-used-by-hackers/
38. https://www.stateoftheinternet.com/types-of-ddos-attacks.html
39. C. Douligeris, A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art", in: Computer Networks, Elsevier, pp: 643–666
40. http://royal.pingdom.com/ 2007/06/01/theoretical-vs-real-world-speed-limit-of-ping/