APPLICATION LOAD SIMULATION AND THE POTENTIAL FOR DENIAL-OF-SERVICE WHEN THE LINUX TOP PROGRAM IS MISUSED

Mark Nordby
St. Cloud State University, MN
noma0401@bcrl.stcloudstate.edu

Sara Krzenski
St. Cloud State University, MN
krsa0601@stcloudstate.edu

Fatma Al Saadi
St. Cloud State University, MN
alfa0401@stcloudstate.edu

Abstract

In computer security, a denial-of-service attack (DoS attack) is a computer crime that violates the Internet proper use policy as indicated by the Internet Architecture Board (IAB) and makes a computer resource unavailable to its intended users. DoS attacks have two general forms. One form causes the victims' computer(s) to reset or consume its resources such that it can no longer provide its intended service. The second form obstructs the communication media between the intended users and the victim in such way that they can no longer communicate adequately. Attacks can be directed at any network device, including attacks on routing devices, Web resources, electronic mail or Domain Name System servers. A DoS attack can be perpetrated in a number of ways.

There are three basic categories of DoS attacks: Protocol attacks, Software Vulnerability Attacks, and Bandwidth/Throughput attacks. In this paper, we will examine a Bandwidth/Throughput attack. This type of attack is caused by consuming the victim's available resources. The attack may consume most of the available network bandwidth shutting out (or significantly delaying) legitimate users, which can result in a loss of business or cause lengthy delays on the network.

With the wider availability and use of free operating systems, the need to establish possible vulnerability thresholds becomes more critical. In our example, we simulate a college campus network using Debian, a distribution of the Linux operating system. Providing students with Linux shell accounts provides them with an open and extremely functional learning environment, but also offers them a powerful platform from which to launch DoS attacks. One possible scenario that can be easily implemented is setting the delay parameter on an interactive command such as top (by default, the refresh rate is 3 seconds). However, it is possible to set it

to sub-second values such as .0001. Therefore, our concern is that students with a very limited knowledge of the LINUX operating systems, using a simple application like top could introduce significant delay on the network and make it difficult for legitimate traffic to reach the campus network.

This paper will look at the effectiveness of a DoS attack using a very simple LINUX program generally available to all users on LINUX platform. Top is a program that will give continual reports about the state of the LINUX system, including a list of the processes using the CPU. In this setup, we have a single host with a progression of up to eight clients. Our first step will be to determine a communication baseline between our host and client 1. We then will use a combination of an increasing number of machines and two possible –d setting for top (-d specifics the delay between screen refreshes). The –d 0.00001 and 0.000001 will be used as our settings for several test runs. We seek to determine if the top command could be used as an effective DoS attack. Our preliminary results have shown that the system handles three attacking clients well, but the network delay increases linearly as each additional attacking client is added with the delay about 4 times greater at the 8 attacking client level than with one attacking client. However, even with 8 attacking clients, the system never completely lockup. Further, the throughput decreased as well, in one case, from ~12,000 bytes per second with one attacking client to ~4,000 bytes per second at the 8 attacking client level.

Keywords: Denial-of-service (DoS), Top command, Throughput attack, Bandwidth attack, Local area network (LAN), TCPdump, TCPstat.

1 Our Concern

With the wider availability and use of free open source operating systems, the need to establish possible thresholds becomes more critical. In our example, we simulate a college campus network using Debian, a Linux distribution operating system. A possible scenario is the students' access to a campus network with many co-located buildings and data centers using standard LINUX user accounts. Therefore, our concern is that students with a very limited knowledge of the LINUX operating system, using a simple application like ***top*** could introduce significant delay on the network and make it difficult for legitimate traffic to reach the campus network.

1.1 Our Study

This paper looks at the effectiveness of a DoS attack using a very simple LINUX program generally available to all users on LINUX platform. ***Top*** is a program that will give continual reports about the state of the LINUX system, including a list of the CPU using processes (2). In this setup we have a single host with a progression of up to eight clients. Our first step was to determine a communication baseline between our host and client 1. We then used a combination of an increasing number of machines and two possible –d setting on ***top*** (-d specifics the delay between screen refreshes) (3). The –d 0.00001 and 0.000001were chosen as our settings after

several test runs. We seek to prove that the ***top*** command could be used as an effective DoS attack. ***Top*** has the potential to use up enough bandwidth to cause a DoS attack on the network resulting in disruption between the host and the other client conversation.

2 Test Simulation: Network Configurations

The basic configuration of the network consists of nine dual core Intel based PCs running Debian, a Linux distribution operating system, connected via a 10/100 network interface to a Force 10, E300 switch (a high end Enterprise level switch with 6 banks of 12, 1Gbs ports). One of these six banks was isolated to simulate a LAN (local area network) so as to eliminate outside interference.

3 Explanation of Experiment

In order to better understand the effect of high intensity applications on a network, ***top*** (a program that provides a look at processor activity in real time and can also be set to precise levels of intensity), was used to simulate a program that required extensive processing power. This ability was crucial to offer a controlled and continuous load that could be measured and compared.

This experiment is run on a private network with Secure Shell as our communication protocol. With the baseline established at each of the two top settings, a new client is added and the test is repeated. This process is duplicated eight times. The goal of the experiment is to create more and more intense traffic levels competing with client1's (our data collection monitor) ability to communicate with the host. We are only concerned with the traffic between the host and client1. TCP Dump was used to trap 100,000 packets in each host client communication session that were used to generate test results.

4 Test Results Analysis

The goal of this paper is to determine the ability to affect a Denial-of-service attack on a client by using the LINUX ***top*** command with increasing number of clients and decreasing delay settings. SAS (*statistical analysis system*) program is used on the data collected using TCPDUMP to generate data work sheets and MS Excel was then used to generate graphs for the purpose of data analysis.

Table 1 and figure 1 represent the total number of bytes against the number of clients. It is clear from the results that at both the 0.00001 and 0.000001 top settings, there is a decline in the number of total bytes transmitted with an increase in the number of clients. Overall, between one and eight clients, there is a performance degradation of approximately 20.23 % (0.00001 setting) and 19.78 % (0.000001 setting). The results show a slight increase in performance with the addition of client 2 (86,244,738 bytes and 87,067,716 bytes respectively) and most of the decline performance initiates on the addition of client 3.

Table 1
Total Bytes Results and Percent Change

top -d Setting in sec	Variable	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client7	Client 8
0.00001	**Total Bytes**	85515812	86244738	70072500	69030604	69246778	64718810	65134136	68211836
	% change with respect to client1	NA	.85239	-18.05901	-19.27738	-19.02459	-24.31948	-23.83380	-20.23483
0.000001	**Total Bytes**	84673960	87067716	71203500	69876392	69797910	68520068	68352432	67924138
	% change with respect to client1	NA	2.83703	-15.90862	-17.47594	-17.56862	-19.07776	-19.27573	-19.78155

Figure 1

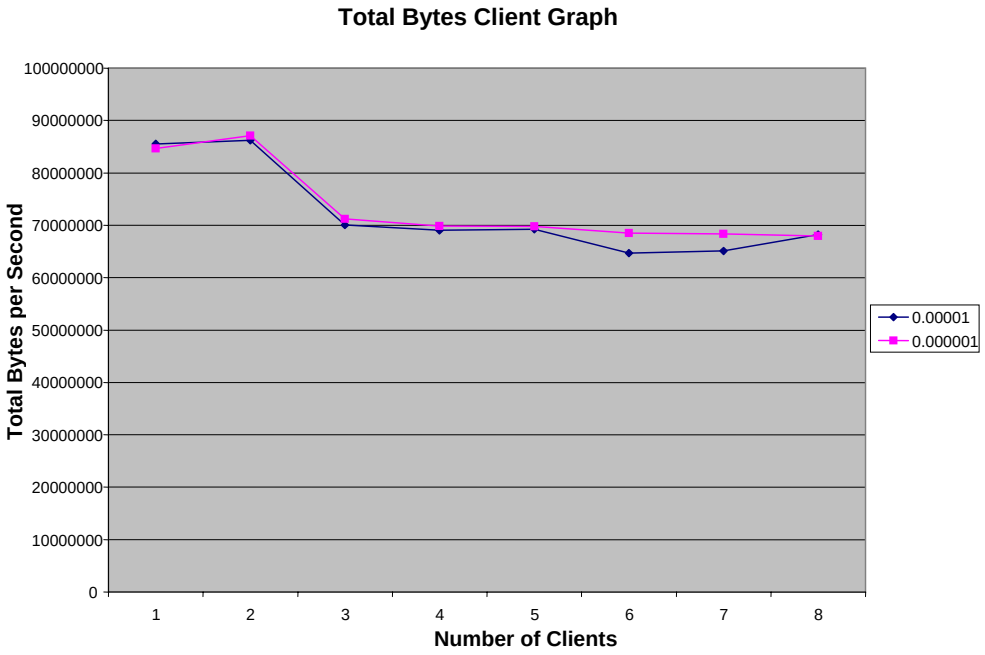


Table 2 depicts the distribution for the mean Inter-Arrival time. The graph (figure2) illustrates an increase in efficiency until the addition of client 3. At client 4, a continuous increase in Inter-Arrival time is observed at both the 0.00001 and 0.000001 second delay settings. This increase in mean Inter-Arrival times for client 1 is to be expected since the server is also in communication with the other clients. The overall percentage deterioration in mean Inter-Arrival times, for both the 0.00001 and 0.000001 second delay settings, is approximately 174.95%. The results confirm the observation that the performance degradation takes place with the addition of client 4.

Table 2
Inter-Arrival Time Descriptive Statistics

Top -d Setting in Sec	Variable	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8
0.00001	N	99501	99501	99501	99501	99501	99501	99501	99501
	Mean IAtime	0.012361	0.011076	0.006428	0.007647	0.007639	0.009082	0.014033	0.017674
	Std Dev	0.009591	0.009745	0.010292	0.012973	0.013021	0.016734	0.027176	0.033444
	Mnimum	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06
	Maximum	0.021775	0.022036	0.049707	0.077517	0.091294	0.144621	0.226811	0.293244
Mean Percentage Change =			174.95%						
0.000001	N	99501	99501	99501	99501	99501	99501	99501	99501
	Mean IAti	0.006934	0.007295	0.006314	0.007806	0.00978	0.011934	0.014479	0.017394
	Std Dev	0.005978	0.007116	0.010368	0.013696	0.017816	0.022072	0.027046	0.032834
	Minimum	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06
	Maximum	0.013602	0.01752	0.080629	0.111438	0.146085	0.178544	0.190163	0.252245

Figure 2
Mean Inter-Arrival time Client Graph

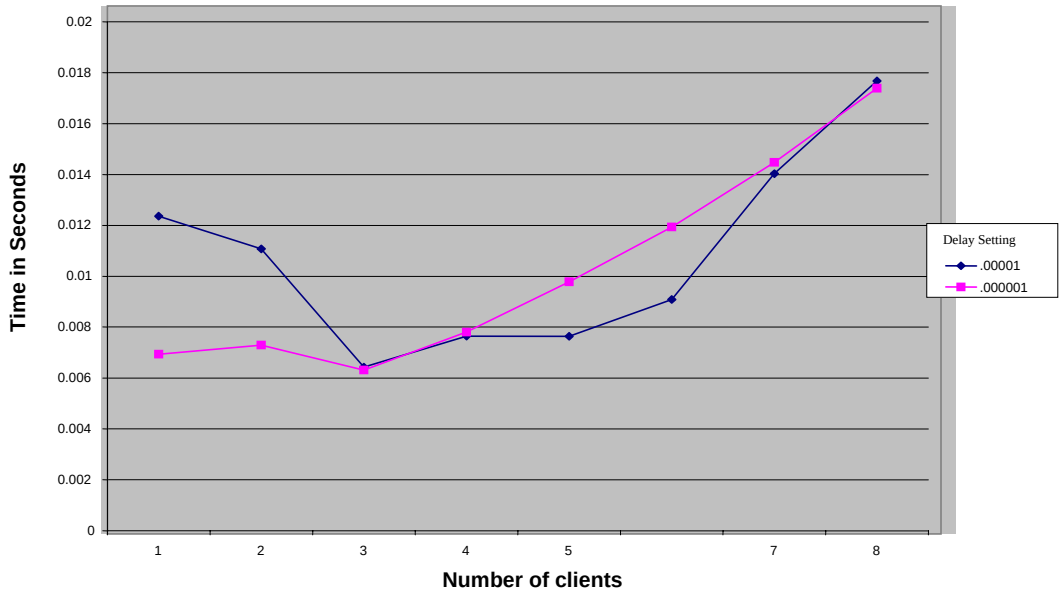
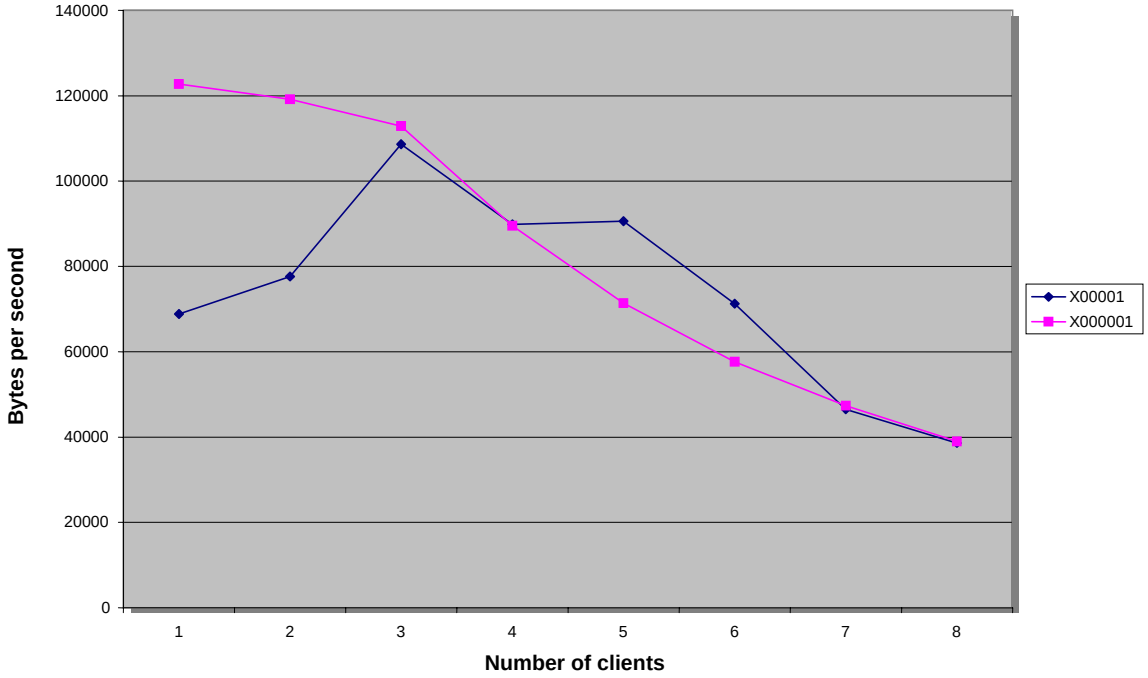


Table 3 shows overall two thirds performance degradation to client 1's communication with the host on running the ***top*** command from eight clients. The one millisecond delay setting results demonstrate a smooth decline in performance in comparison to that of 10 milliseconds. The results show no performance difference between both the delay settings on the addition of client 7 and client 8. Given more time, it would have been interesting to observe the performance at smaller delay settings.

Table 3
Mean Throughput Results and Percent Change

top -d Setting in Sec	Variable	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8
0.00001	Mean Throughput in B/Sec	68860.33	77652.04	108637	89827.02	90623.2	71240.95	46500.04	38589.25
	% Change with Respect to Client 1	NA	11.32193	57.76427	30.44814	31.60436	3.45717	-32.47194	-43.96011
0.000001	Mean Throughput in B/Sec	122713.4	119181.9	112869.8	89503.14	71391.68	57607.07	47378.7	38964.03
	% Change with Respect to Client 1	NA	-2.87784	-8.02162	-27.06327	-41.82243	-53.05560	-61.39077	-68.24794

Figure3
Mean Throughput Client Graph

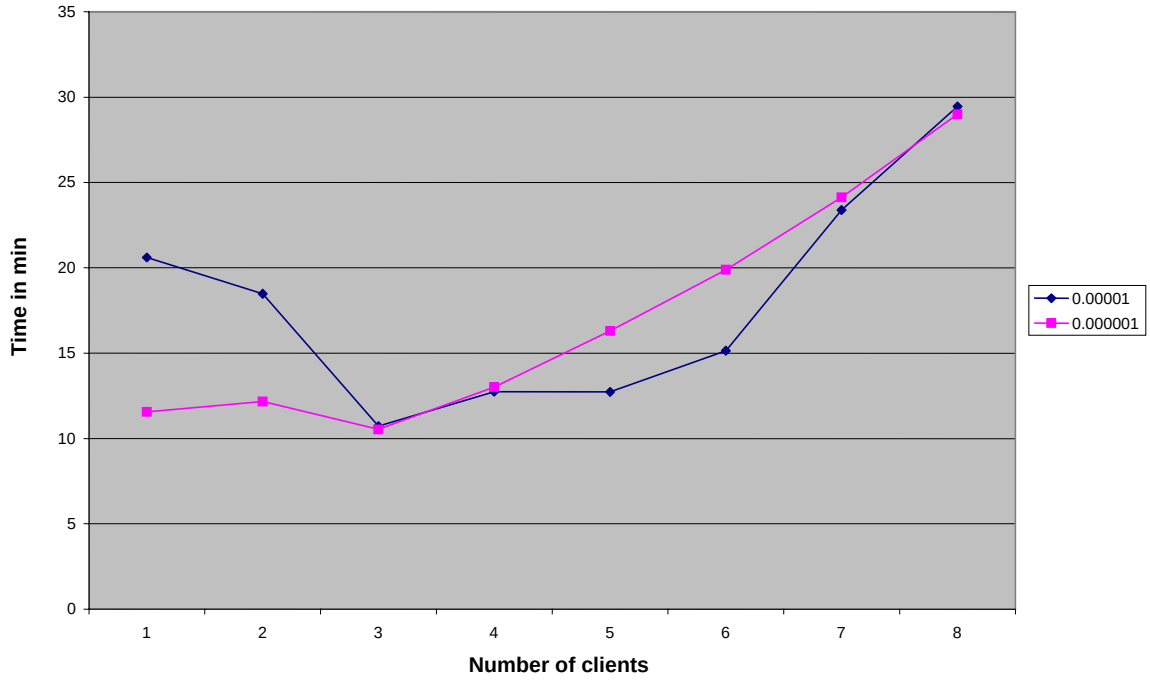


As predicted the total time for trapping 100,000 packets increases for client 1 with the increase in the number of clients communicating with the host. Furthermore, as observed earlier at client 3 maximum efficiency is achieved. After client 4 performance degradation occurs. Here again the results show almost two thirds overall performance degradation as is depicted in Table 4 and Figure 4.

Table 4
Total time and percent change

Setting in sec	Variable	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8
0.00001	Total time in min	20.60506	18.47166	10.71526	12.74468	12.73114	15.13713	23.38294	29.4553
	% change with respect to client 1	NA	-9.8898	-47.99694	-38.14781	-38.21353	-26.53683	13.48154	42.95178
0.000001	Total time in min	11.55656	12.15783	10.52507	13.00605	16.30187	19.88411	24.12416	28.99239
	respect to client 1	NA	5.20285	-8.92558	12.54257	41.06161	72.05907	108.74862	150.87388

Figure 4
Total Time Client Graph



5 Conclusions and Recommendations

The results show that there was a performance loss (approximately two thirds) when the workload was increased, however it was shown that at this number of clients, the use of the TOP program was not enough too generate enough throughput that would cause a denial of service occurrence.

A couple of factors may have contributed to the results of this test. First the test was ran with multiple established connections using SSH (Secure Shell) from clients to host. This unlike a typical application that uses TCP/IP maintains a continuous connection to the host thereby not allowing more communication downtime, which would thus reduce bandwidth consumption. Second, the workload that was generated although considered a heavy load on a production machine may have not been as much of a factor due to the fact that the test unit was stripped down and running only the minimal applications thus having less information to update each time TOP refreshed the display.

Due to the results of this paper more research needs to be done on the possible effects other applications may have on network traffic when proper monitoring of application usage could be overlooked.

Works Cited

1. Moshe Benyamini, Ori Modai, Tzach Schechner, Yaniv Stern. (2001). "DdoS Project Final Report". http://www.comnet.technion.ac.il/~cn2wo3/final2.doc
2. Linux/ Unix Command: Top. http://linux.about.com/od/commands/l/blcmdl1_top.htm
3. Michael Palmer, Jack Dent, Tony Gaddis, (2005). Guide to UNIX Using Linux 3rd ed. Canada: Thomson Course Technology.

References

Dennis Guster, Abdullah AlHamamah, Paul Safonov, (2002), Network Security Log Information: A Preliminary Analysis. Business Computing Research Center College of Business St. Cloud State University

Dennis Guster, Renat Sultanov, Mark Nordby, Richard Sundheim, (2004), Using Distributed Processing to Enhance Performance Characteristics of Hosts Used to Support WWW Applications. Business Computing Research Laboratory, St. Cloud State University