Monitoring and Managing Cloud Computing Security using Denial of Service Bandwidth Allowance

Biswajit Panja
University of Michigan-Flint, Flint, MI 48502
bpanja@umich.edu

Bharat Bhargava
Purdue University
West Lafayette, IN 47907
bb@cs.purdue.edu

Sourav Pati
University of Michigan-Flint, Flint, MI 48502
spati@umflint.edu

Dayton Paul
University of Michigan-Flint, Flint, MI 48502
daytonp@umflint.edu

Leszek T. Lilien
Western Michigan University
Kalamazoo, MI 49008
llilien@wmich.edu

Priyanka Meharia
Eastern Michigan University, Ypsilanti, MI 48197
pmeharia@emich.edu

***Abstract*— Over the next decade, cloud computing has a good chance of becoming a widely used technology. However, many challenges face the cloud to be overcome before the average user or business team will trust their vital information with a cloud server. Most of these challenges tie into developing sound security measures for the cloud. One of the largest security obstacles is how to defend against a Denial-of-Service (DOS) or Distributed Denial-of-Service (DDOS) attacks from taking down a cloud server. DOS attacks are nothing new; many strategies have been proposed and tested against DOS attacks on networks. However, none have been able to completely prevent DOS attacks. The search continues for an effective solution to keep data available to legitimate users who need it when the cloud network that stores that data is the target of a DOS attack. The method proposed (DOSBAD) in this paper will explain how effectively detecting the bandwidth limit of a cloud network and the bandwidth currently in use to know when a DOS is beginning.**

Keywords- Cloud computing; Denial of service; Bandwidth

I. INTRODUCTION

It is believed that the world is heading towards a computing resource grid similar to the power grid and being charged based on usage like we are for energy[FZ 08]. There are similarities and differences between grid computing and cloud computing [FZ 08]. They share a lot in common, but the cloud is less secure though it utilizes virtualization. The future of computing may be centralized around cloud computing, and client computing for. People and organization may want to do computing on cloud instead of investing resources locally and implement security in their local computers. That leads the focus on providing security in cloud. Cloud computing needs set standards, has layered architecture (software as a service, platform as a service, infrastructure as a service, and hardware as a service), the different modes of clouds (private, public, and hybrid), types of virtualization used in cloud computing (server, storage, and network virtualization), fault tolerance, security issues, and scalability.

In this paper we propose an approach to avoid Denial of Service (DOS) attacks. In this, an entity integrated into a cloud server can be used to monitor what ratio of available bandwidth is being used. To find the maximum available bandwidth of the server, the entity DOSBAD (Denial-of-Service-Bandwidth-Allowance-Device), will periodically send a series of packets down each possible path within the cloud (router-to-router). Two large packets are first sent to create a queue at the switch between the routers, then two small packets are sent, which will be the ones that have their time of being sent and their time of being received measured. The total time to transfer these packets will be the time at which packet 1 is sent subtracted from the time at which packet 2 is received. Based on the time it takes for the receiver to receive the packets and to acknowledge them, DOSBAD will calculate the bandwidth available between those two routers. DOSBAD will also monitor how much of that bandwidth is in use at each router. The number of incoming packets will be measured, along with the amount of acknowledgement packets that are sent back out. Ideally, the number of packets received should match the number of acknowledgement packets sent back out, indicating that the router is not overwhelmed with the number of incoming packets. When the number of incoming packets starts to outweigh the number of acknowledgement packets sent, that can indicate that the bandwidth limit may be close to being reached. This indicates that either there is an abnormal spike in activity coming into the network (i.e. a flash crowd), or that there is malicious activity being attempted. At this point, DOSBAD may look for common return addresses on incoming packets at the overwhelmed router(s) and then send out a ping to those addresses. If DOSBAD does not receive a response from an address, that may indicate a DOS attack being attempted. DOSBAD signals to the router (or possibly the gateway) to drop all incoming packets from that address. Another feature that the cloud manager may wish to use is to have DOSBAD automatically change the address of the attacked router so that if the attacker tries again from a different attacking address, he is unable to find that router again. All other legitimate traffic will be re-routed to the new address automatically.

DOSBAD keeps a running tab on the addresses of all senders of incoming packets within some time interval. DOSBAD uses this to see from which address the most incoming packets are coming from. Along with this can be stored the signature of each incoming packet. The signatures of packets coming from zombies in a DOS attack sometimes have very specific signatures that can be used to detect that a DOS attack is occurring. If a high ratio of bandwidth is being used, one or more routers are overwhelmed by incoming packets, and a high number of packets are coming in at a router from the same IP address, DOSBAD will proceed to investigate a possible DOS attack by pinging the suspicious address or addresses as mentioned previously.

This paper is organized as follows, section II provides related work. Section III describes the architecture of

DOSBAD. Section IV provides the proposed security approach for cloud computing.

II. Related Work

Dikaiakos et al. [DK 09] talks about the types of things that are expected for the future of cloud computing. Included in this are ideas of infrastructures, platforms, and software being offered as services which are bought by "consumers" or anyone who wants to implement the services. Clouds have some characteristics that help us describe their type, including "internal" or "external/hosted", and "private", "public", or "hybrid". The three layers of a cloud are infrastructure (lowest level), platform (higher abstraction), and application (highest abstraction, provides actual applications that consumers can buy). Also discussed are the challenges that must be solved in order to realize the full potential of the cloud, including the architecture of the cloud versus individual computers, data management and security, cloud interoperability (customers using the cloud applications through different types of machines), and the economics involved with purchasing services.

Armbrust et al [MF 09] starts off by defining what the Cloud is. Their definition is that the Cloud itself is the hardware and software that is needed to provide services of a network to many Cloud users. Cloud providers provide the resources to Cloud users who implement the resources to create applications that Software-as-a-Service users can use for what is called Cloud Computing. Another aspect discussed is the reasons why the Cloud is taking off now and not previously. These reasons include the quick, low-commitment services to users such as PayPal and real-time responsive applications.

The three classes of utility computing, which is what Cloud users purchase from providers, are defined by 3 different abstraction levels for resources provided to Cloud users. For low-level abstraction the user has more flexibility with what kinds of applications they want to program but limit the scalability of the application is very limited (it's hard to change the limits on the application if the demand for it suddenly skyrockets above the set limit). For high-level abstraction the user can make things that are much more scalable but not very flexible for general computing since the user cannot control the low-level hardware. Mid-level abstractions provide some aspects of the previous two classes. General-purpose computing and multiple programming languages are available (low-level) and the libraries help provide limited scalability (high-level). Each of these classes have different models for how they provide computations, storage, and networking to users. For now, none of the three classes have proven to be the most useful out of the three. Each of them is ideal for certain situations.

The next aspect discussed is Cloud economics. The decision of whether to host a service through the Cloud or to continue using datacenter can be answered by looking at several things. If your average utilization and peak utilization very different values, that is a reason for switching to the Cloud because the scalability of resources can help a host not have to pay for unused resources during non-peak times. Another aspect to consider is the cost of transferring all of the user's data from their datacenter to the Cloud: will the time saved by doing so outweigh this cost? If so, the Cloud could be a realistic option. Also, the heating costs saved from using the virtual machines of the Cloud to provide services can be a positive to switching to the Cloud.

The final section of the article goes over ten of the largest obstacles that Cloud Computing must overcome and ten corresponding opportunities that can be used to overcome these obstacles. Obstacle one is the possibility that demand for Cloud Computing could overtake the practical supply of resources that a Cloud provider can meet. The solution discussed is how this encourages a cooperative effort among multiple companies to greatly increase supply. Obstacle two involves service users being locked in to one Cloud user (the one that provides the services that buys resources from the Cloud provider), which can be solved by standardizing services among all Cloud users so that people looking to buy a service can choose who to buy from. Obstacle three involves data on the cloud being secure enough; solutions include things we already implement with networks, such as encryption and firewalls. The fourth through eighth obstacles involve how the Cloud will grow over time, such as data transfer bottlenecks, performance unpredictability, and bugs in the distributed systems. Solutions to these include physically shipping the disks to save money, implementing flash memory to reduce the interrupts and thus increasing performance, and creating a debugger that works with virtual machines (respective to the three mentioned obstacles). The last two obstacles look at the business aspect of the Cloud. The first one is how to prevent reputation fate sharing from a few bad users (spammers, etc.). The concept of trusted email services that already exist could be applied to help guard the reputation of services. Finally, software licensing can cause a problem because a user could purchase a service and not be able to use that service on other computers. The solution is to offer pay-as-you-go options so that the user can pay for what they need as they realize they need it.

In the conclusion of the article, specific implementations of applications, infrastructure, and hardware are mentioned that should be implemented in future systems to be more easily Cloud compatible. Applications should be able to run partly in the client and partly in the Cloud, each part having its own duties. Infrastructures should be designed to run on virtual machines. Hardware systems need to be designed as containers instead of single boxes or racks since users will purchase them in containers. Besides this, questions are posed to the reader as to what the future Cloud systems will be like.

Vouk et al. [VM 04] defines the concept of cloud computing, describes various aspects involved with cloud computing, an

example implementation of cloud computing at North Carolina State University, and lastly about research issues involved with the cloud.

Cloud computing can be considered the next step in improving the availability of services and products supplied to users over a network that is in part due to virtualizing the resources. One aspect mentioned that is crucial to clouds is the service-oriented architecture, which means users request services from the cloud provider. Another critical aspect is making services out of components, which can be described by reusability, substitutability, extensibility, scalability, customizability, composability, reliability, availability, and security. A workflow can be used to visually represent services that may be provided by the cloud, usually through a graph. A question posed by the article to the reader is whether or not workflows could be useful in representing the infrastructure of cloud computing. Another aspect discussed related to cloud computing is virtualization of various computing components, such as memory, hardware, and applications. Cloud computing relies heavily on virtualization because it allows computing components to become more portable so they can be provided to users easily as a service. The final and most important aspect of cloud computing are the four types of users this article defines that are involved with the cloud. These are developers, who configure and maintain the Cloud framework, service authors, who develop templates for services from the Cloud framework, service composition experts, who create services for end-users, and end-users, who request services and implement them. The final topic of this article is the research issues of cloud computing, including getting feedback on workflows, collecting, storing, and preventing provenance information, optimization of service components, service portability, cloud computing security, and efficient utilizations of resources.

Mowbray et al. [MP 09] goes into some specific privacy issues with regards to cloud computing and defines one possible solution Privacy Manager that could be used to overcome these issues. The main requirements defined here include minimizing the user's data stored on the cloud to what is necessity, protection to what data is stored on the cloud, limiting the purposes that can use the data and the people who may access the data, user-controlled preferences related to what their data may be used for on the cloud, and feedback given to users about how their information was used afterward.

The solution this article offers to meet the requirements is called Privacy Manager. It uses five main features to both protect user data and give the user a welcoming sense that he or she has control of their own data (customizable features). One of the biggest features is obfuscation of data that goes into the cloud and de-obfuscation of data that is being accessed by the user from the cloud. Obfuscation is similar to encryption, only the user gets to decide on a specific key that is used to modify their data as it goes into the cloud. The key is not provided to the cloud provider so that they cannot de-obfuscate the data themselves. With preference setting, another feature, the user can decide what data gets obfuscated and what data doesn't (sometimes you don't want to obfuscate data for certain applications). The data access feature allows users to see what data they have on the cloud to make sure it is accurate. If the data is obfuscated in the cloud, it gets de-obfuscated by the Privacy manager before being shown to the user. The feedback feature shows the user how their data is being used in the cloud (so that the user will know if their preferences have been violated). Finally, the personae feature can be used so that a user can set up different levels of preferences with different cloud services (obfuscating some information when using certain applications and not obfuscating the same information for other applications). They simply choose the personae that has the preferences they want to apply to the current application.

Cho et al [CB 11] explain with large programs often have a daunting number of lines (usually millions). It is very difficult to track down all the bugs in such a program that could be used by someone malicious to, for instance, initiate a denial-of-service attack or cause a segmentation fault and wreaking havoc. Finding these errors by just using software to send it random input data to check the resulting output does not always find all these errors, as demonstrated by an experiment in this article. A solution this article offers is a new approach to exploring behavior of programs given vast varieties of input data called MACE: Model-inference-Assisted Concolic Exploration.

MACE consists of sending messages to an algorithm called L*. L* infers a state machine based on this input. For every state in this finite state machine (Mealy Machine), L* generates a path to get to that state that is the shortest possible path (i.e. if you want to reach a certain state S, it finds the shortest length of input string that will take you to that state). One input string is used per unique state so that all states can be analyzed using state-space exploration. The output from each of these states plus the input used into L* in the first place are sent through a filter to get rid of redundant inputs and a new list is sent to L* to make a new FSM, and the process keeps going until no new states are found through an iteration.

An experiment was ran to test MACE against a baseline method of analyzing programs using the state-space exploration part of MACE without using the component that sends the input back to the L* algorithm. The baseline method uncovered only one vulnerability in the programs tested (Vino and Samba) while MACE found seven between the two, four of which had never been discovered before that on record. MACE also generated a fairly accurate FSM of Vino compared to what it actually was. Other comparisons include the number of detected crashes (30 to 20), unique crashes (9 to 1), and the exploration depth, which showed MACE was more proficient at reaching deeper states than the baseline approach.

Some limitations of MACE are discussed after the experiment. It cannot be guaranteed that MACE found ever possible vulnerability because of how the L* algorithm works. Also, MACE was very good at analyzing user-level programs, but it was not able to go into the kernel level, which limits its effectiveness.

The conclusion poses some questions for the reader to think about for future experiments, including how FSMs can be studied further to find even more effective implementations within the state-space exploration component that MACE uses. Another question is how to find a better way to filter out redundant output messages without eliminating possible new states and thus finding new vulnerabilities. The third question is if there are any other feasible methods besides using FSMs to help generate all possible output sequences for a program.

III. ARCHITECTURE OF DOSBAD

Cloud computing is an indispensable part of the software world; it is believed that the world is heading up to the computation of software services as an unit .According to a group presentation by Chrissy Hanlon et all cloud computing is a model which was conceived in 1961 by John McCarthy who dreamed of a computing service as an utility. As a matter of fact this model is widely used in today's era, and although it has been widely used by public and organizations, every day a fresh news or blog items alarms us about its security issues. Security threats seem to have presented a major hurdle to the wide acceptance of cloud computing. According to *Bernd Grobaur et al Siemens*, security is cited as the substantial roadblock for cloud computing –uptake. Some example of security issues are, denial of services attack, side channel attack, authentication attack and much other type of security threats. These attacks are catastrophic therefore defense is indispensable, in fact many organizations have invested enormous effort for finding an efficient defense, there are many solutions have been found such as *DDOS defense as a network service by Ping Du et al, Implementing Pushback: Router-Based Defense Against DDoS Attacks et al ,AT&T Lab Research*, but certain flaws in those solutions still thwarting fuller implementation of cloud computing by everyone, be it end user, large organization or small business .As a repercussion of this lurking security vulnerability, cloud computing has still been unable to reach every nook and cranny of the software world.

Among of all the aforementioned security threats, we choose to work on DOS attacks as it is the most common and very serious threats mentioned by other researcher such as *Bernd Grobaur et al Siemens, AT&T Lab Research*. In addition, it is agreed that DOS attacks are very difficult to defend against as they do not target any specific vulnerability of systems but rather the target is to any device connected to network.On one hand developing DOS attack is an easy task as there are many user friendly tools are available. On the other hand, developing a robust solution for preventing DOS attack is a taxing job and yet to manifest. In this paper we will discuss several experiments aimed at protecting against DOS attacks. One of the experiments we implemented is called denial of services by bandwidth allowance-device (DOSBAD); this is a procedure by which we are trying to detect the attacker. To accomplish this goal we have used the following specifications:-

- Windows 2007
- Programing language Java 1.7
- Swing

In the programming, we have implemented a model view controller (MVC-2) architecture where we can assume the view is accessible to any number of user and the view is playing the role of client side machine, we have a java class called **ProtectedServerThread** which is basically acting as a server, in the experiment we are accessing packets from the server and rendering it to the *end user*. More details about the architecture and technical specification is given in the experiment and implementation section .In our experiment we are actually implementing a concept similar to the clad method developed byAkhirio Nako et al.

In this paper we would explain a novel use of bandwidth allowance which would track the bandwidth and allow packets to receive only if the bandwidth is under a threshold level; beyond the threshold level no packets are accepted. As we mentioned earlier that two different experiments were implemented and each of them are having different procedure but the architecture is same as by both the experiment we are trying to protect the attack based on the available bandwidth, in addition the experiment also draw a graph on transaction time for each events .

In this section we discuss the architecture of our proposed model DOSBAD.

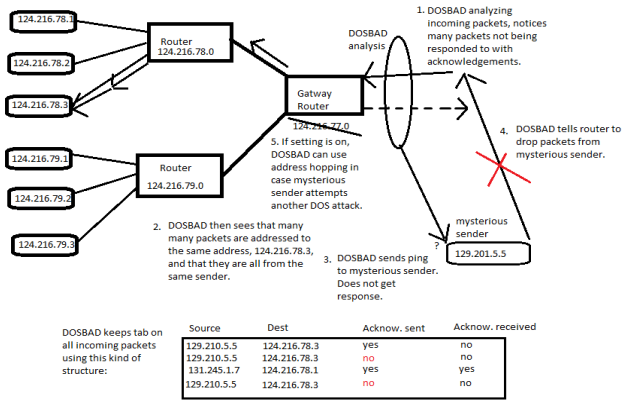


Figure 1: Architecture of DOSBAD

DOSBAD periodically measures the available bandwidth along the paths of the network. It does this using a variant of the probe gap model discussed in Huan Liu's article, "A New Form of DOS Attack in a Cloud and Its Avoidance

Mechanism. With this variant, more than just a pair of packets is used. A series of 1500 byte packets are sent along the desired measuring path in order to create a queue at the router. Then two 64 byte packets are sent down the path. The time it takes for the second 64 byte packet to reach the receiver is used with the equation:

$$\text{Available bandwidth} = C * (1 - \frac{\Delta o - t_p}{\Delta i})$$

Where C is the maximum bandwidth of the path, Δo is the time gap that the receiver measures between receiving each of the 64 byte packets, t_p is the time to transmit the second packet, and Δi is the time gap between sending each of the 64 byte packets.

The packets should usually be sent along the narrowest path in terms of bandwidth in the network, since that is the most vulnerable area in the network. Repeating the packet sending process once every second, a trend can be observed as to if the available bandwidth begins to change drastically. Also, if the available bandwidth falls to a certain amount to inhibit network performance, DOSBAD can detect this and investigate possible causes.

At some ratio of available bandwidth to maximum bandwidth B, the network will become sluggish. If DOSBAD detects a B at or less than this ratio, it begins looking through its traffic list. The way DOSBAD stores traffic information looks something like this:

Source (32 bits)	Destination (32 bits)	Acknowledgement Sent (0 for no, 1 for yes) (1 bit)	Acknowledgement Received (0 or 1) (1 bit)	Duration in milliseconds (up to 1 second) (10 bits)
129.210.5.5	124.216.78.3	0	0	301
129.210.5.5	124.216.78.3	0	0	498
131.245.1.7	124.216.78.1	1	1	543
129.210.5.5	124.216.78.3	1	0	782

DOSBAD stores each instance of traffic going through the network, either to or from a host within the network, within the last second. When DOSBAD detects low available bandwidth, DOSBAD can check this dynamic table, checking for many instances of the same source or destination address. In this rather simple example, there are many instances of the IP address 129.210.5.5 sending packets to the destination within the network of 124.216.78.3. We see that an acknowledgement was sent out for the first instance of traffic, but was not received, and that the other instances from this IP address were not even sent an acknowledgement. This means that 129.210.5.5 is the most likely suspect of launching a DOS attack if there is one being launched.

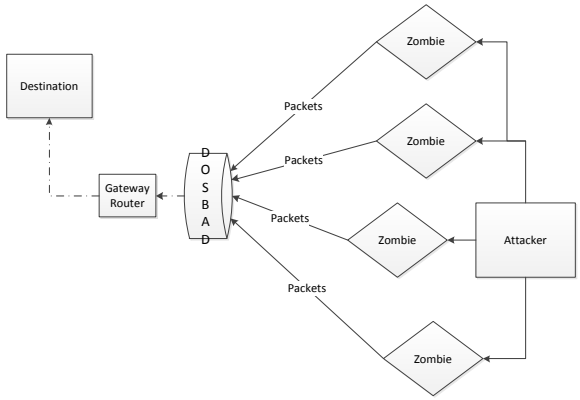


Figure 2: Attack Model

Normally, all incoming packets are going to be encrypted, so DOSBAD cannot check the packets itself to see if the packets look valid from the content. DOSBAD must therefore use packet signature authentication on the packets coming from the suspicious IP address. DOSBAD will store a list of known signatures, much like an antivirus program, and compare the incoming signatures to this list. If they find a match, the means the packet is part of a DOS attack and DOSBAD can have the incoming packets from that IP address dropped.

It may not always be the case that a perpetrator's IP address can be identified. Some attackers spoof their IP address, or use zombie machines to launch a distributed DOS attack. In that case, the only way DOSBAD has to detect the attack is to look at only the destination IP address within the network that has the most packets being sent to it:

Source (32 bits)	Destination (32 bits)	Acknowledgement Sent (0 for no, 1 for yes) (1 bit)	Acknowledgement Received (0 or 1) (1 bit)	Duration in milliseconds (up to 1 second) (10 bits)
127.215.17.2	124.216.78.3	0	0	301
125.127.18.1	124.216.78.3	0	0	498
131.245.1.7	124.216.78.1	1	1	543
125.117.21.4	124.216.78.3	1	0	782

Again, DOSBAD notices the many packets being sent to 124.216.78.3. DOSBAD will also still notice the unreturned acknowledgements, even though all the source IP addresses are different. This can indicate a distributed DOS attack against the network. Since it isn't as simple as just dropping the packets from a specific source, DOSBAD will have to check for a 1 on the acknowledgement sent bit with a 0 on the acknowledgement received bit. This ensures that DOSBAD is dealing with one of the zombies since they won't return the acknowledgement. DOSBAD again uses packet signature analysis by comparing the signature of the incoming packet with its list of known attack signatures. Upon finding a match, DOSBAD will again know for sure that this packet is part of a DOS or DDOS. 125.117.21.4 packets will be dropped, then their instances will leave the table. In an updated table, the 1 bit for acknowledgement sent will now move to a different

zombie, since the destination has now moved on to trying to verify a different sender. DOSBAD can then check for another 1 0 combination on those two bits and then have the gateway drop that address. This process may continue for a while until network performance returns to normal.

If performance is so bad that no legitimate traffic is getting through at all, it may be beneficial to implement application hopping. The services provided by the destination may be temporarily moved to a different host that isn't receiving nearly as much traffic until the bandwidth to the burdened host can be freed up. This setting is customizable by the cloud service provider.

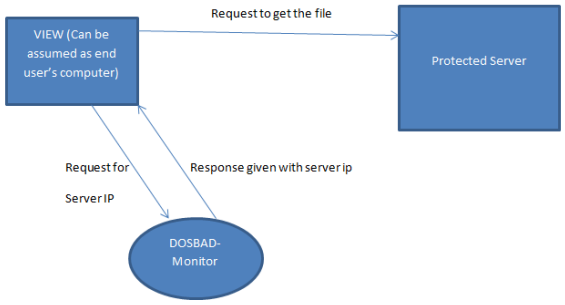
A situation may occur where DOSBAD's list of known signatures is not up to date with every possible attack signature. In this case, DOSBAD will not be able to detect a signature that is not on its list. With this situation, it may be helpful to log the IP address that DOSBAD does packet signature analysis on, use application hopping, and then using the log to see if there is a new attack signature that can be derived and added to the list of known signatures.

IV. Experiments and Implementation

We have implemented a model view controller (MVC-2) architecture where we can assume the view is accessible to any number of user and the view is playing the role of a client side machine, we have a java class called ProtectedServerThread which is basically acting as a server, in the experiment we are accessing packets from the server and rendering it to the *end user*.

There are two type of experiment we have implemented so far and for each experiment we tried to follow the same architecture, we design a cloud attack and tried to identify the response time of the system under the attack as well as when free from any attack. In one experiment DOSBAD works as a Monitor, DOSBAD has many java objects which can be viewed as different virtual machine. We have a thread which is basically acting as protected server therefore we would be referring this thread as "Thread Server" throughout the paper while explaining the experiment details. In this experiment we track the response time and transaction time. The architecture is a model view controller (MVC). Although it's MVC pattern is implemented in java technology but we are assuming the view as client side machine and our multiple instances of the view can be assumed as multiple clients' machine from which a request is supposed to be sent. Architecturally, DOSBAD instances works as a network service running on cloud infrastructures.

In the experiment 2 we treated DOSBAD as a monitor and in this experiment we have multiple threads which are basically events. In this experiment we are trying to identify the response time for multiple events happening at the same time which conceptually gives the same environment when a server is under attack. Below is the figure of the architecture of the experiment 2



Experiment: I

In the experimentI it is discussed a server's response under attack. We have created multiple threads which we are considering as number of events are fired by the end user. In order to achieve this we used a java class. The java class works in the steps given below. (1) Firstly it checks the current available bandwidth. We have defined an available bandwidth and threshold bandwidth for our testing purpose. This value can be changed based on the environment.

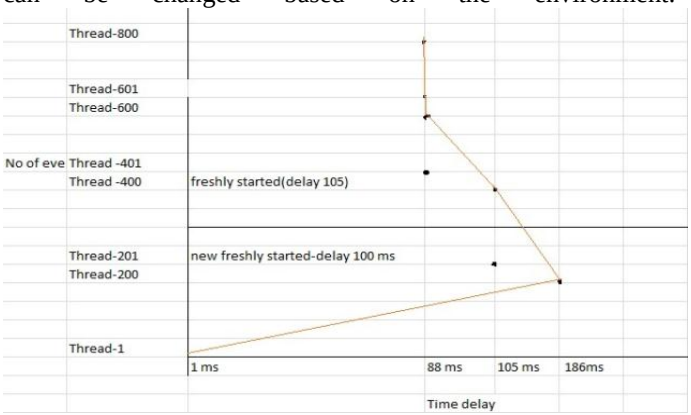


Fig 3

The program is designed in such a way that if the current available bandwidth is below threshold label bandwidth then the user (in this case the event) is allowed to get into the server and access the packets and render it to the end user . This experiment is made easy to understand in such a way that we just print the packet details in the console). So the question is what happens when the current available bandwidth is more than the threshold label? This where our program is classifying a malicious attack .On account of visualizing the response time by the server, we temporarily allowed the program to give access to the server and let the sever send the response with the packet details .However, in this case we are tracing the time the time delay between every events (in practical this would be the delay between different users). The response time delay for each event is shown in the graph Figure 3. The graph is drawn between number of events and time delay to get the response from the server for each event. The time delay for the first event was 1 ms and then for next 200 hundreds events the time delay grows till 186 ms. At this stage it is conspicuous that there was a long time gap between Thread 200 and Thread 201 is 100 ms . For The second slot of 200 events the server started working normal and time delay

started decreasing towards 88 ms afterword's the response time became constant till 800 events.
What we are achieving by this experiment is, with reference to the other experiments such as cladder we saw that they are trying to identify the ip address and based on that they are identifying the attacker, but as we know that making replica of ip address is quite easy in today's era; so our solution might be the one which would be more robust as the detection happens in the programing and also it checks the available bandwidth which can be not duplicate.

Mathetical expression for the experiment I
Assumption
Ab-Available bandwidth
Number of Threads are X;
Eu- End user
Tb=Time before execution
Ta =time after execution,
Tt= transaction time
D= delay

When $Ab < Tb$ then Eu is valid. If $Ab > Tb$ then Eu is invalid.

Transaction time for each thread $Tt = Ta - Tb$

Total delay D = transaction time for each event $\times$ number of events

Hence $D = X(Ta - Tb)$ or $D = X \times Tt$ (millisecond)

Assuming number of threads = T
And time delay = t
Increase in thread= ΔT and in change in time = Δt [$\Delta \rightarrow$ difference]
From the graph we can substitute the value of ΔT = Thread1 to Thread200 = number of thread is 200
Hence $\Delta T = 200$
The response time for 200 threads is 1ms to 186ms
$\therefore \Delta T = (186 - 1) = 185ms$
$\therefore \frac{increase\ in\ threads}{change\ in\ time} = \frac{\Delta T}{\Delta t} = \frac{200}{185}$

Hence to derive the equation we can write

$$\frac{dT}{dt} = \frac{200}{185} = 1.08$$

From the above calculation we can prove that the derivative of thread and time will always have a +ve data, which means the graph will be growing upon increasing the number of threads. Therefore there will be always time delay under the attack.

Experiment II:
In this experiment the implementation done with a broader detail. We have implemented graphical user interface using swing technology. Our assumption is the GUI is running on a sever which can be accessed from any place in the world. In the experiment we have a protected server which is basically java class .We assumed that the *ProtectedThreadServer* is the server placed in a descent distant .In order to achieve the experiment successful MVC-2 architecture is implemented .The model view controller (MVC-2) works in such a way that the view which is developed using java swing creates a graphical user interface and given to the end user. An end user can access a file contains some information, the file is located in the server. However, to get the access of the server the end user needs to get permission from the DOSABD.In this experiment the DOSBAD is like a monitor, playing the role of a mediator who takes the user information along with the bandwidth and test the current bandwidth and available bandwidth .if the available band width is lesser than the threshold bandwidth then the DOSBAND provides the protected server ip address to the end user. So in short the end user sends a request which firstly goes to the DOSBAD monitor , the DOSBAD monitor check the user details and identify the user based on the current available bandwidth and the threshold bandwidth ,on one hand if the user is suspected the DOSBAD does not allow the end user to hit the "Get Server " button which can give access to the server and the server would return a file contains some information.The figure 4 shows that DOSBAD has found the user as attacker therefore the "Get Server " button is disabled. On the other hand, once the user is identified as legitimate the DOSBAD monitor sends the protected server's ip address to the end user and enable the "Get Server" button as shown in the figure 5. At this point the end user is allowed to click on the get server button as shown in the below figure. In comparison with the aforementioned first experiment, in this experiment also we are trying to identify the transaction time for each request.

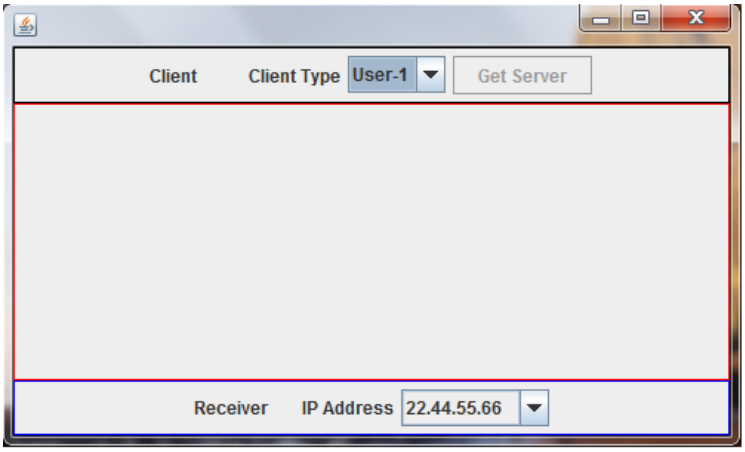


Figure 4

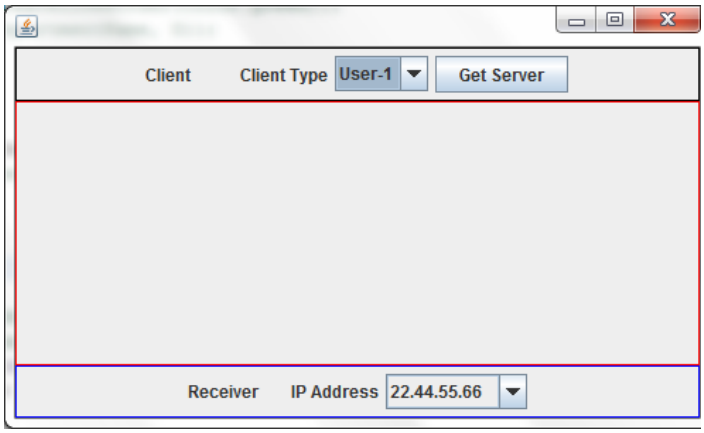


Figure 5

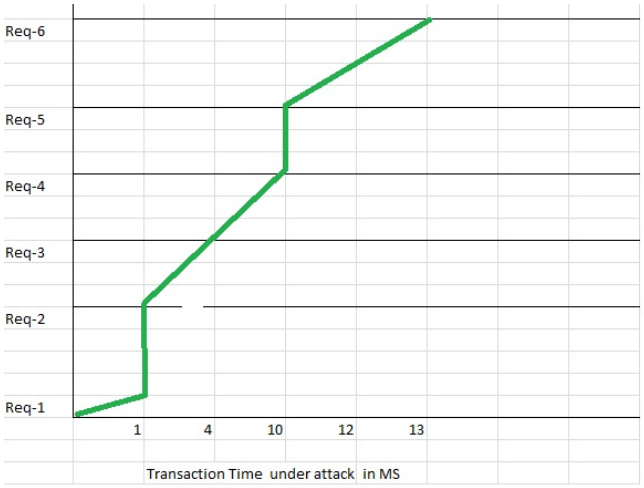


Figure 6

The graph in the Figure-6 shows the transaction time by the server under the attack. As mentioned, this graph is drawn considering the server is under attack, when the attack takes place the java program dynamically creates 100 requests and each request goes to the server to collect the file and return to the end user, in the graph in figure 6 it shows the transaction time that is the time to collect the file and print the file content of the file in the console. For each request the transaction time grows high. For Request -1 and 2 transactions time is 1 ms and for Request 3- Request4 and the transaction time grows higher upon increment of number of request.

The experiment has sought to introduce a defense against Denial of service attack using the DOSBAD monitor's novel functionality i.e. tracking using the current available and threshold label bandwidth. This solution is immensely cost-efficient as the tracking would take place by a java program; the program needs to be connected to the device from which the bandwidth would be available. Most current DOS defense provides implementation by end-host or ISPs which are not very cost-efficient especially for small companies. We believe our design to be offered as a new solution for the dangerous DOS attack and save the cloud computing model from jeopardy.

Consider the above graph for the experiment II
Assume no of request is R and time = T
Now increase in request = ΔR and change in time = ΔT [Δ→ difference]

Assume initial value of R1= 1 and R5= 5 (no of request cant not be –ve value)

Hence $\Delta R = R5 - R1 = 4 \quad equation\ 1$

Similarly from the graph T1= 1 sec and T4=10 sec

$\therefore \Delta T = T4 - T1 = 4\ ms\ equation\ 2$

$\therefore \frac{incease\ in\ rerqest}{change\ in\ time} = \frac{\Delta R}{\Delta T}$ or assuming difference in request and difference in time is very small.

$\Delta R/\Delta T = \frac{dr}{dt}$ substituting the value obtained from the above equation 1 and equation 2

Hence $\frac{dr}{dt} = \frac{4}{9} = 0.44$

Therefore using the above equation we can prove that upon increasing number of request the graph will always have growth and consequently there will be always a time delay under the attack.

V. Conclusion and Future Work

In this paper we propose a protocol(DOSBAD) to avoid Denial of Service (DOS) attacks in cloud servers. DOSBAD integrated into a cloud server can be used to monitor what ratio of available bandwidth is being used. To find the maximum available bandwidth of the server, DOSBAD periodically send a series of packets down each possible path within the cloud (router-to-router).

This protocol can be improved by implementing it in actual cloud servers. Different DOS or DDOS attacks can simulated to make sure it can handle multiple attacks at the same time.

References

[FZ 08] Cloud Computing and Grid Computing 360-Degree Compared Foster, I.; Yong Zhao; Raicu, I.; Lu, S.; Grid Computing Environments Workshop, 2008. GCE '08

[DK 09] Dikaiakos, M.D.; Katsaros, D.; Mehra, P.; Pallis, G.; Vakali, A.; Cloud Computing: Distributed Internet Computing for IT and Scientific Research Internet Computing, IEEE 2009

[MF 09] M Armbrust, A Fox, R Griffith, AD Joseph, RH Katz. Above the Clouds: A Berkeley View of Cloud Computing - 2009, UC Berkley

[VM 04] MA Vouk - Cloud Computing – Issues, Research, and Implementations, Journal of Computing and Information Technology, 2004.

[MP 09]Miranda Mowbray, Siani Pearson, A Client-Based Privacy Manager for Cloud Computing, COMSWARE '09 Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middleware

[CB 11] Chia Yuan Cho, Domagoj Babic, Pongsin Poosankam, Kevin Zhijie Chen, Dawn Song and Edward XueJun Wu, "MACE:

Model-inference-Assisted Concolic Exploration for Protocol and Vulnerability Discovery", To appear in Proceedings of the 20th USENIX Security Symposium, (USENIX Security'11)

[MB 08] Johns, M.; Engelmann, B.; Posegga, J. XSSDS: Server-side Detection of Cross-site Scripting Attacks, Computer Security Applications Conference, 2008. ACSAC 2008.

[WJ 11] Jansen, W.A.; Cloud Hooks: Security and Privacy Issues in Cloud Computing System Sciences (HICSS), 2011 44th Hawaii International Conference on

[XG 09] Jinpeng Wei Xiaolan Zhang Glenn Ammons Vasanth Bala Peng Ning, Managing Security of Virtual Machine Images in a Cloud Environment,CCSW '09 Proceedings of the 2009 ACM workshop on Cloud computing security

[BC 09] Rimal, B.P.; Eunmi Choi; Lumb, I.; A Taxonomy and Survey of Cloud Computing Systems, INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference

[FY 08] Foster, I.; Yong Zhao; Raicu, I.; Lu, S.; Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop, 2008. GCE '08

[BG 09] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis, Energy-Efficient Cloud Computing, Incorporating Special Issue: Architecture/OS Support for Embedded Multi-Core Systems, 2009

[WW 09] Cong Wang; Qian Wang; Kui Ren; Wenjing Lou; Ensuring Data Storage Security in Cloud Computing, Quality of Service, 2009. IWQoS. 17th International Workshop

[WW 11] Cong Wang; Qian Wang; Kui Ren; Wenjing Lou , Improved Verifiability Scheme for Data Storage in Cloud Computing, Wuhan University Journal of Natural Sciences 2011

[XB 06] Wei Xu , Eep Bhatkar , R. Sekar, Practical Dynamic Taint Analysis for Countering Input Validation Attacks on Web Applications, 15th USENIX Security Symposium (Vancouver, BC, Canada, August 2006).

[ZS 09] Xinwen Zhang, Joshua Schiffman, Simon Gibbs, Anugeetha Kunjithapatham, Sangoh Jeong, Securing Elastic Applications on Mobile Devices for Cloud Computin, CCSW '09 Proceedings of the 2009 ACM workshop on Cloud computing security

[YR 09] Liang Yan, Chunming Rong and Gansen Zhao, Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography, Cloud Computing Lecture Notes in Computer Science, 2009

[WS 02] Anthony D.Wood, John A. Stankovic,Denial of service in Sensor network, University of Virginia

[SN 07] Lakshmi Santhanam, Deepti Nandiraju, Nagesh Nandiraju and Dharma P. Agrawal, Active Cache Based Defense against DoS Attacks in Wireless Mesh Network,University of Cincinaty

[MP 02] Jelena Mirkovi´c Gregory Prier Peter Reiher, Attacking DDoS at the Source_ University of California Los Angeles

[DM] *Christos Douligeris and Aikaterini Mitrokotsa*, DDOS ATTACKS *AND* DEFENSE MECHANISMS: A CLASSIFICATION University of Piraeus, Piraeus, Greece

[LP 08] Ming Luo, Tao Peng, Christopher Leckie, CPU-based DoS Attacks Against SIP Servers,The university of Melbourne