Distributed Denial of Service attack on Cloud: Detection and Prevention

Bikram Khadka[1], Chandana Withana[1], Abeer Alsadoon[1],Amr Elchouemi[2]

[1]School of Computing and Mathematics, Charles Sturt University, Sydney, Australia
[2]Hewlett Packard

Abstract- **Cloud computing is a distributive and scalable computing architecture. It provides sharing of data and other resources which are accessible from any part of the world for a very low cost. However, Security is one major concern for such computing environment. Distributed Denial of Service (DDoS) is an attack that consumes all the cloud resources may have making it unavailable to other general users. This paper identifies characteristics of DDoS attack and provides an Intrusion Detection System (IDS) tool based on Snort to detect DDoS. The proposed tool will alert the network administrator regarding any attack for any possible resources and the nature of the attack. Also, it suspends the attacker for some time to allow the network admin to implement a fall back plan. As Snort is an open source system, modifying different parameters of the system showed a significant aid in not only detection of DDoS, but also reduction the time for the down time of the network. The proposed tool helps minimize the effect of DDoS by detecting the attack at very early stage and by altering with various parameters which facilitates easy diagnose of the problem.**

Keywords—cloud computing; security; DDoS; snort; open-source

I. INTRODUCTION (*HEADING 1*)

Cloud computing is the best way to share data and resources in the field of computing today. It has gained its popularity due its nature of scalability and for being out of physical boundaries. It has become rule of thumb not only because it offers greater flexibility, but because of the different services it provides on different levels [1]. One advantage of Cloud computing is that user had to pay for using cloud services as much as they use. This could increase or decrease the volume of resources according to their budget. This makes cloud computing accessible to users of all categories.

Cloud computing offers different services on different levels, called levels of abstraction, they are: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). SaaS is the end user side of cloud computing. It offers users with online information sharing and storage across the globe. Google doc is an example of SaaS. In PaaS, service providers like Google engine abstracts the machine instance and other technical details from the developer [2]. PaaS is what made the local host into the cloud. IaaS is the most basic level of abstraction that has dedicated servers controlled by the developers. This computing model is for developers to write their software programs over an infrastructure. According to context where the cloud computing is used, it is categorized as Private, Public and Hybrid cloud. Private cloud is created and controlled within an enterprise; were the private clouds exist within an enterprise firewall. The major point is, the company is in charge of deploying and maintaining private cloud services. In case of public cloud, an organization selling cloud services controls public clouds. The public cloud services are available to general public. A third party organization sells the cloud service and general public uses the service remotely on pay-per-use basis. On the other hand, hybrid cloud is combination of both public and private cloud; it uses services that are in both public and private cloud domain. Control and maintenance responsibilities are shared between the cloud provider and the business itself.

Although cloud computing has numbers of benefits to offer to its users, it also has some challenges associated to its use. Security in cloud computing is the most serious and discussed topic these days [3]. The cloud computing is in danger of several security threats. Distributed denial of service attack (DDoS) is one of the attacks where the attacker consumes every resources the server has until the server literally becomes unavailable [4]. A review of different security literatures on cloud computing reveal that in spite of great attention paid to it, no significant solution has been devised yet. Earlier research on cloud security issue proposed a solution to tackle DoS [3]. It suggested the use of threshold method to detect intrusion into the cloud server. In threshold method, when access to a particular IP address exceeds an arbitrary number, a threshold, would be labeled as an attack and blocked afterwards. Implementation showed that the threshold could be guessed and this firewall against DDoS could easily be bypassed. Additional research on load balancing as a means to defeat DDoS was conducted [5]. This solution was more of hybrid firewalling technique where the end focus was to mitigate the effect once DDoS hits the server. The proposed method worked well on physical layer but it lacked the support for application layer.

The purpose of this paper is to contribute to the theoretical understanding of DDoS behavior and come up with an effective solution to it. In the quest of the effective solution, this paper also introduces snort, free and open source software. The reason to choose free and open source software is to make the solution more accessible and less costly. Another good thing about snort is, it has huge user community to support. The research tunes various components of snort and modifies it to suite the exact requirement to counterpart the DDoS attack. It then implements the proposed solution in a virtual environment to test its effectiveness and usefulness. The remainder of the paper is structured as follows: First, a thorough literature review would be presented in section 2 followed by descriptions of proposed model in section 3. The proposed work implemented and results were discussed in section 4. The paper concludes with a summary of the study's research contribution and directions for future research.

II. PREVIOUS WORKS

A. Framework and standardization based solution

Cryptographic storage and network firewalls can somehow tackle DDoS [1]. It studies the effect on different network anomaly-based attack detection techniques under cloud data migration. A toolchain method is used to experiment the detection performance of two anomaly detection techniques; Principal Component Analysis (PCA) and Expectation-Maximization (EM). A toolchain simulates attacks and virtual service migration. The study then records the performance of the given detection technique to identify which one is legitimate request and which one is an attack. The study reveals that in few setting, some malicious attacks are missed and unnecessarily, high number of alarms pertaining normal behavior were generated. In case of PCA, clustering with and without migration were recorded. Half of the alarms generated were for non-malicious requests which the study says a false alarm. Likewise, EM tends to be failed to classify between malicious and non-malicious attack. The solution concludes stating these anomaly detection techniques unreliable in case of virtual service migration. Although the paper labeled these algorithms as unreliable for virtual migration, it does not state why they fail to play their role. Apart from the security, quality of service is also important in the event of attack. Since an attack gradually degrades the quality of service that the cloud usually provides, it should also be considered [6].

Yet another attempt to deal with DDoS has been done by Dubey, Bhajia [7]. It has detailed analysis of what DoS attack is and what happens during the attack. Also, it talks about the additional service usage charge that the end users have to pay because of Dos attack. It then comes up with firewalling and IDS as an immediate solution to it. Although the proposed solution is not a novel solution, it can be an immediate countermeasure to DDoS. A strategy to reduce DDoS attack by implementing hybrid cloud based firewalling architecture is proposed by [5]. The solution identifies different kinds of DDoS attack like ICMP flood, which is also known as ping of death or Smurf and the SYN flood attack. The ICMP flood attack has impact on not only the victim but the whole network. It triggers when an attacker sends huge amounts of echo-reply packets. On the other hand, the SYN flood attack uses TCP three-way-handshake to increase the number of half open connection and saturate the network resources of the server. Rather than prevention, the solution is more effect mitigation oriented. It offers a hybrid cloud based firewalling architecture to reduce the impact of DDoS once it hits the server. The theme of this architecture is the collaboration of physical and virtual firewall through a communication module. When a physical firewall is overloaded, the traffic is redirected to virtual firewall. This is done by a component called Virtual Firewall Management Unit, VMFU, of the architecture. The solution presents two scenarios of DDoS attack and experiments the performance of the server under attack. Also, it records the performance of the server when the proposed architecture is implemented in order to facilitate comparison. However, the proposed solution has only been experimented for physical layer; it has no evidence of working the same over the application layer.

B. Method based solution

To tackle DDoS Sharifi et al. [8] proposed a model. The solution employs Intruder Detection System (IDS). Once the number of request exceeds the threshold, IDS assumes that an attack blocks the user. However, danger of false prediction to penetrate through the IDS remains. The solution also features load balancing and honey pot models. In load balancing, different servers share loads which protect any one server from being overwhelmed by the request load. Likewise, this is unlikely to work in a large DDoS attack where the attack occupies all the servers that are available. A honeypot misguides the attacker as a vulnerable point in the system. It then records all the data that goes in and out which might not be available with IDS. Because an attacker can find out the weak points in the system themselves, there is no guarantee that the honeypot suits best to gather all information regarding the attack.

A proposal to use randomized encryption key to tackle DDoS like the cloud security issue has been proposed in [9]. The proposal is devised to be able to give better performance than encryption based algorithms such as Prediction Based Encryption (PBE) and Identity Based Encryption (IBE). According to this algorithm, a series of cipher text are generated at first. Then one of the cipher texts is chosen on a random basis to pair with the original plain text. Although it is a good thing to choose a random key out of the generated cipher keys, a problem of cipher text occurs as it is longer than the original plain. At the receivers end, the encrypted random key is decrypted with the shared key and then encrypted data is decrypted with decrypted random key. The idea behind this algorithm is the generation and use of the random key to encrypt and decrypt. Two levels of encryption can be seen in this algorithm. First with the key, which is generated randomly, second the data which is encrypted again with the encrypted random key. The solution is not experimented and doesn't give any experiment that can ascertain about the strength of the proposed solution. Storage efficiency is somehow related to DDoS since everything it does, it does by consuming servers' resources. In this scenario, cloud computing faces storage efficiency challenges [10].

A threshold method to identify DDoS attack is presented in [3]. The philosophy of this solution is if the user exceeds the number of requests vs. the set threshold value, then the user is treated as an attacker and blocked. It doesn't just set a random value as a threshold; it has a method to determine it. Number of source IP packet is tracked from server's log. If a particular IP is found more than *n* consecutive packet sizes than normal over a given period, then the IP is labeled as an attacker and blocked. This solution however suffers from the problem of false detection of the threshold value. Also, the time frame where the IP on the server's log is observed does not have a standard value.

Thorough review of literature on DDoS states that there has not been found a prominent solution yet. Researchers have found various techniques to prevent DDoS to some extent and mitigate effect once it hits the server at any point. Cloud, in particular, due to its scalability, can be the most impacted by DDoS [11]. It means, when DDoS hits one server, in the cloud computing environment, another server jumps in to share the load. This means, to share the attack. Eventually almost all servers get involved into this DDoS, making the entire network victim of one single DDoS. Nonetheless, improvements in IT security schemes and findings of new countermeasures are drilling down DDoS further down.

III. PROPOSED MODEL

Despite all restless efforts made to solve DDoS problem from the root, a single standard solution has not been found yet. This paper proposes a solution to DDoS on the basis of a tool. A tool for Intrusion Detection (ID) based on free and open source software, snort[11], will be implemented. As stated earlier, snort, on its own, is a base file containing list of rules for network traffic monitoring. Support users of the snort community regularly update the rule chain to cope with new circumstances. In order to add value to the network monitoring activity, which is very crucial in terms of DDoS, the model also uses WireShark [12]. A WireShark is also a freely available software package that comes handy when gathering network related information.

ID systems are placed in a very specific area of the network. Some believe it should be placed between the internal firewall of a system and the Wide Area Network (WAN) [4]. This is so because when a request packet comes inside the network boundary, first the deployed firewall plays its role. Even if the infected packet manages to get through the firewall, an ID will be there to alert the network administrator with bit of possible information regarding the infected data packet. There are different methods to send the alert message such as in console, sending text message, sending email and sending pop-ups. For snort to work in this particular environment, tuning of certain variable is necessary. Variables like defining domain, internal/external network range and writing some custom configuration and rule files.

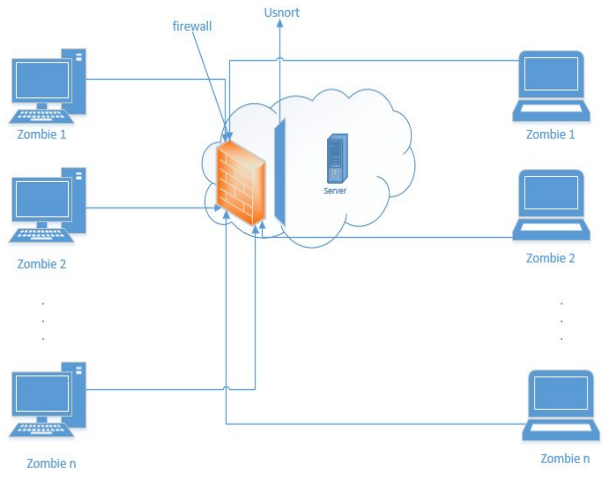


Fig. 1. Model of Usnort architecture

Figure 1 above illustrates the working model of Usnort. The tool is an updated version of snort hence named Updated snort (Usnort). As the nature of the attack is distributed, there are attackers in different geographic boundaries. Since the attackers behave like normal users and attack afterwards, they are named zombies. Different zombies are assumed to be on different geographic locations to simulate the distributed environment. As shown in the figure 1, when a attacker tries to penetrate through the server, the first thing it has to go through will be the firewall. The firewall is implemented to protect the system from unwanted and malicious attacks. As the attacking system has become more advanced, the can somehow bypass the firewall and tries to get into the server. There comes Usnort. As soon as the criteria for snort meets for a user request to be attack, Usnort does two things; First, it alerts the network administrator about the intrusion with all relevant data associated with the attack such as source IP, destination IP, source port, destination port, payload size etc. secondly, it drops the connection with that IP for a certain time being so that the network administrator can start the fall back plan. In this way the server can be prepared before the attack actually starts infecting it.

Furthermore, one interesting feature of Usnort is, it logs the whole sequence if communication between client (attacker) and the server. The logs are very important to diagnose the problem. The log basically will have all the details of the client data packet that is not shown in the alert message. It can be handy in the event of SYN Flood attack. In SYN Flood attack, the client sends a Hello to the server. In response to this, server also sends an ACK message acknowledging the client that the Hello has been received by the server and it can start sending further data on provided SEQ (Sequence) number. A genuine user would then start sending its data on the given SEQ number but, attacker won't do that. The attacker ignores the ACK message sent by the server and keeps on sending the same Hello message again and again. The server assumes that the client never received the ACK it sent and it also sends ACK again and again. When this sequence continues for lots of packets, server is literally occupied and cannot server further use. In such this event, Usnort logs every detail, including the client-server handshaking. It can record when the server received Hello from client and when it sent its ACK. Since the information is provided in sequence on regular time frame, one can easily understand that there has been an attempt to SYN Flood attack.

Another feature upon which Usnort identifies attack is through TCP payload. A TCP payload is a part of IP data packet. It is calculated as follows:

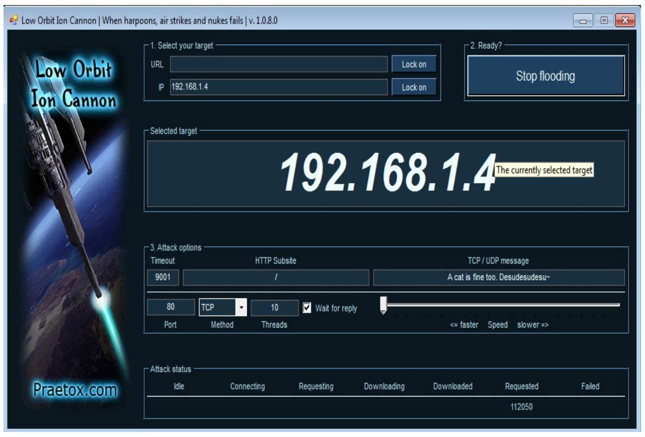
TCP payload = ip.len-ip.hdr_len-tcp_hdr_len).

Given the standard size of TCP payload 1460 bytes, Usnort has a defined rule to flag an alarm when it receives a TCP payload greater than 1460 bytes. This is because every normal TCP packet is equal or below the standard size and when it contains malicious codes, it is always greater than standard size. In Usnort's log, such communication is marked with a different color which can be seen with any log editor file like WireShark. This feature provides one more aid in prevention of DDoS attack.

IV. IMPLEMENTATION/METHODOLOGY

To simulate a distributed nature of DDoS, virtual environment is created. On that virtual machine [13], there is a server (cloud server) and some clients. The clients do not receive IP from DHCP. This is to show that they are located in different geographical regions. Usnort is installed on the server and placed between the firewall zone of the internal server and the external wide area network. Low Orbit Ion Cannon (LOIC) is freely available software to test DDoS. It needs the source URL or the source IP, port address and service, whether TCP, UDP or HTTP, to attack. Once all relevant information is given, LOIC starts sending huge amount of request with the chosen service to the server. WireShark is used to capture network relevant data and to read the log file in this solution. On Usnort's configuration file, home net is set as the network address of the server and optionally all its domain. External net is configured as anything that is not home net.
In order to start the implantation, two scenarios have been developed. First, an attack is done with the use of LOIC using TCP as a service and port 80 as target botnet (vulnerable port) without the user of Usnort. Server's performance is monitored throughout the attack. In the next scenario, the same attacking procedure is followed having Usnort implemented. Server's performance is measured again while Usnort is implemented.

A. *Scenario 1: without Usnort*



IP 192.168.1.4

192.168.1.4

Timeout 9001 | HTTP Subsite / | TCP / UDP message A cat is fine too. Desudesudesu~

Port 80 | Method TCP | Threads 10

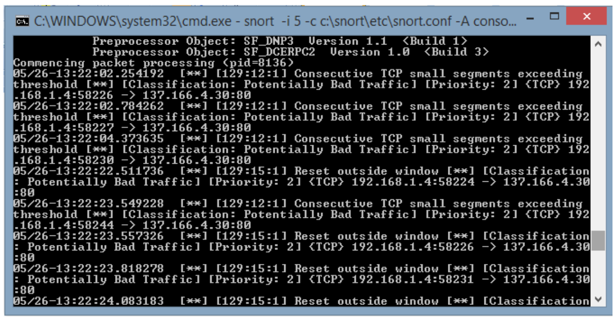
Requested 112050

Fig. 2. Screen capture of DDoS attack testing tool LOIC

Parameters for the attack are set such as IP address of the target server, service and port number. Refer to the figure 2, on the bottom right corner, the numbers of packet flooding is shown, which could only be seen while attack. The network admin has nothing to monitor over the command line interface. He doesn't even know that the server is under attack

B. *Scenario 2: Usnort implemented*

Under the same TCP flooding attack, with Usnort implemented, the network admin can see alert message coming through with all possible relevant information of the attack.



```
C:\WINDOWS\system32\cmd.exe - snort  -i 5 -c c:\snort\etc\snort.conf -A conso...
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=8136)
05/26-13:22:02.254192  [**] [129:12:1] Consecutive TCP small segments exceeding
threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192
.168.1.4:58226 -> 137.166.4.30:80
05/26-13:22:02.784262  [**] [129:12:1] Consecutive TCP small segments exceeding
threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192
.168.1.4:58227 -> 137.166.4.30:80
05/26-13:22:04.373635  [**] [129:12:1] Consecutive TCP small segments exceeding
threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192
.168.1.4:58230 -> 137.166.4.30:80
05/26-13:22:22.511736  [**] [129:15:1] Reset outside window [**] [Classification
: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.1.4:58224 -> 137.166.4.30
:80
05/26-13:22:23.549228  [**] [129:12:1] Consecutive TCP small segments exceeding
threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192
.168.1.4:58244 -> 137.166.4.30:80
05/26-13:22:23.557326  [**] [129:15:1] Reset outside window [**] [Classification
: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.1.4:58226 -> 137.166.4.30
:80
05/26-13:22:23.818278  [**] [129:15:1] Reset outside window [**] [Classification
: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.1.4:58231 -> 137.166.4.30
:80
05/26-13:22:24.083183  [**] [129:15:1] Reset outside window [**] [Classification
```

Fig. 3. command screen scenario 2

As per the figure 3, there are alert messages which the network admin can see very easily. It has information regarding which service is on target, what port and what the source is and destination IP addresses. One point to notice here is the message that says "Reset outside window". It is a unique feature that Usnort provides. What it means is, it has dropped connection with the source IP address for certain time. This gives the network administrator enough time to diagnose and start the fall back plan. A quick demo on snort config file that does this is written on the snort rules. It is written as: "drop TCP any any > any 80 (\

msg:"Reset outside window"; \

detection_filter:track by_src, count 30, seconds 1; \

new_action drop; timeout 50; sid:1000001;)"

In the rule, a TCP service over port 80 is being observed. The rule is written to drop a TCP request on port 80 when it exceeds 30 requests in 1 second. When this rule triggers, it throws a message saying it has reset outside the window, this is a proof that the rule has been executed and worked. Timeout simply means till what time the new connection attempt by the source IP will be avoided. In this case, the attacker IP cannot establish any connection with the server for 50 seconds. 'Sid' is a unique ID for each rule file associated with some registered rule events. The 'Sid' up to 100000 is reserved for the special purpose. So, any custom rule needs to have 'Sid' greater than 100000 and no two rule can have same 'Sid'.

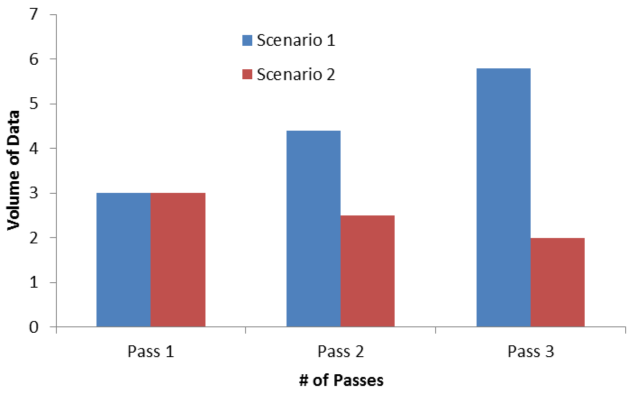


Fig. 4. Effect Analysis

In both the scenarios, same volume of attack is applied. As can be seen from the bar chart above in figure 4, the results are tested and analyzed using different passes. Pass 1, marks the entry of an attack where both the scenarios are on the same state. The resource consumption, illustrated over the vertical axis of the chart, increases in pass 2 for scenario 1, whereas for scenario 2 it gets down. The same pattern continues over in pass 3. When IDS is not implemented, the attack keeps on penetrating and the resource consumption keeps on going high. In contrast to this, scenario 2, realizes significant improvement over the volume of attack.

V. Results and Discussions

As from scenario 2, it is clear that the use of Usnort showed the alert of potentially bad traffic going on around the network. Here are two cases of CPU performance under attacks; One when Usnort is not implemented and another when Usnort is implemented (Figure 5).

As can be seen from the figure 5, the CPU resources get consumed by DDoS gradually from 1 to over 25 and then 30. The consumption of CPU resource gradually increases until the server gets literally down.

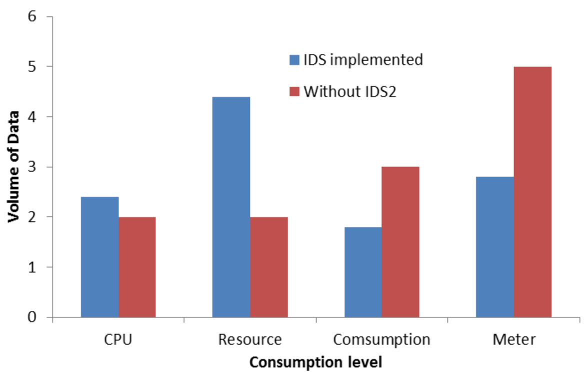
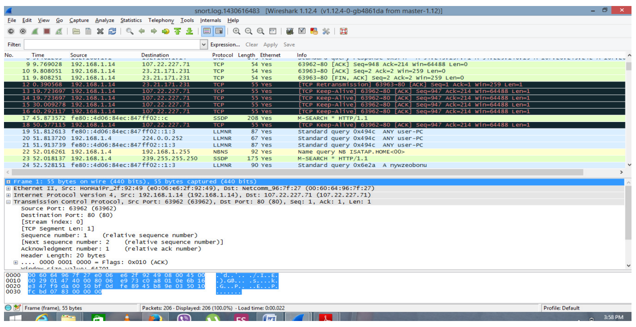


Fig. 5. CPU monitor under DDoS attack

When Usnort is implanted, first time when the attack takes place, the resource consumption of the server is high. But when it starts taking action, it frees up the resources and takes it back to near normal. It is obvious that the resource consumption of CPU gets to its maximum point under the influence of DDoS attack. After that point, it keeps on attacking the server until when the server stops responding. It can be seen with the experiment that the resource consumption gets higher and higher according to the degree of attack. But, with Usnort implemented, it prevents the attack to some extent and gives the network admin enough time to take necessary action. The resource consumption of CPU starts getting down enabling it to serve other general users. Few false alarms were generated followed by the attack. Usnort treated normal user requests as in intrusion when tested immediately after the attack.


snort.log.1430616483 [Wireshark 1.12.4 (v1.12.4-0-gb4861da from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time      Source                  Destination      Protocol Length Ethernet Info
 9 9.769028    192.168.1.14            107.22.227.71    TCP       54 Yes  63962→80 [ACK] Seq=948 Ack=214 Win=64488 Len=0
10 9.808051    192.168.1.14            23.21.171.231    TCP       54 Yes  63963→80 [ACK] Seq=2 Ack=2 Win=259 Len=0
11 9.808251    192.168.1.14            23.21.171.231    TCP       54 Yes  63963→80 [FIN, ACK] Seq=2 Ack=2 Win=259 Len=0
12 0.390568    192.168.1.14            23.21.171.231    TCP       55 Yes  [TCP Retransmission] 63963→80 [ACK] Seq=1 Ack=1 Win=259 Len=1
13 19.723697   192.168.1.14            107.22.227.71    TCP       55 Yes  [TCP Keep-Alive] 63962→80 [ACK] Seq=947 Ack=214 Win=64488 Len=1
14 19.723697   192.168.1.14            107.22.227.71    TCP       55 Yes  [TCP Keep-Alive] 63962→80 [ACK] Seq=947 Ack=214 Win=64488 Len=1
15 30.009278   192.168.1.14            107.22.227.71    TCP       55 Yes  [TCP Keep-Alive] 63962→80 [ACK] Seq=947 Ack=214 Win=64488 Len=1
16 40.292117   192.168.1.14            107.22.227.71    TCP       55 Yes  [TCP Keep-Alive] 63962→80 [ACK] Seq=947 Ack=214 Win=64488 Len=1
17 45.873572   fe80::4d06:84ec:847     ff02::c          SSDP     208 Yes  M-SEARCH * HTTP/1.1
18 50.577115   192.168.1.14            107.22.227.71    TCP       55 Yes  [TCP Keep-Alive] 63962→80 [ACK] Seq=947 Ack=214 Win=64488 Len=1
19 51.812613   fe80::4d06:84ec:847     ff02::1:3        LLMNR     87 Yes  Standard query 0x494c  ANY user-PC
20 51.813720   192.168.1.4             224.0.0.252      LLMNR     67 Yes  Standard query 0x494c  ANY user-PC
21 51.913739   fe80::4d06:84ec:847     ff02::1:3        LLMNR     87 Yes  Standard query 0x494c  ANY user-PC
22 52.016261   192.168.1.4             192.168.1.255    NBNS      92 Yes  Name query NB ISATAP.HOME<00>
23 52.018137   192.168.1.4             239.255.255.250  SSDP     175 Yes  M-SEARCH * HTTP/1.1
24 52.528151   fe80::4d06:84ec:847     ff02::1:3        LLMNR     90 Yes  Standard query 0x6e2a  A nywzeobonu

⊞ Frame 1: 55 bytes on wire (440 bits), 55 bytes captured (440 bits)
⊞ Ethernet II, Src: HonHaiPr_2f:92:49 (e0:06:e6:2f:92:49), Dst: Netcomm_96:7f:27 (00:60:64:96:7f:27)
⊞ Internet Protocol Version 4, Src: 192.168.1.14 (192.168.1.14), Dst: 107.22.227.71 (107.22.227.71)
⊟ Transmission Control Protocol, Src Port: 63962 (63962), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1
    Source Port: 63962 (63962)
    Destination Port: 80 (80)
    [Stream index: 0]
    [TCP Segment Len: 1]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 2    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 20 bytes
  ⊞ .... 0000 0001 0000 = Flags: 0x010 (ACK)

0000  00 60 64 96 7f 27 e0 06  e6 2f 92 49 08 00 45 00   .`d..'.. ./.I..E.
0010  00 29 01 47 40 00 80 06  e9 73 c0 a8 01 0e 6b 16   .).G@... .s....k.
0020  e3 47 f9 da 00 50 bf 0d  fe 89 45 b8 9e 03 50 10   .G...P.. ..E...P.
0030  fc bd 07 83 00 00 00                               .......

Frame (frame), 55 bytes   Packets: 206 · Displayed: 206 (100.0%) · Load time: 0:00.022   Profile: Default
3:58 PM
```

Fig. 6. WireShark log viewer

Figure 6 is a screen capture of WireShark that is used to capture the network traffic and view the log file. Opened is a log file when DDoS attacked the server. The log file contains every single details of the event which is very crucial to diagnose the problem. Information such as sequence number, source IP, destination IP, protocol used, length of TCP header, whether Ethernet or not, and additional info are summarized in the log file. There are different options to log any event. The log can be on binary format, printable format or combination of both. In binary format, details are summarized as binary information and especial binary file editor is required to view such files. Printable format of log is what shown on the figure above. This version is suited for easy reading purpose.

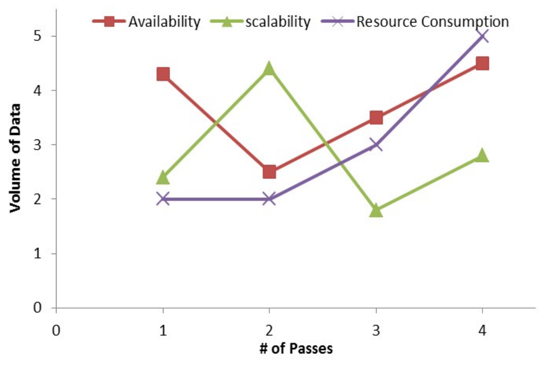


Fig. 7. Analysis under attack

The chart in figure 7 shows different aspects of a cloud server under DDoS attack. Series of attacks are done via different passes, and volume of attacks is measured along Y-axis. It can be seen that, when an attack hits the server, its availability is under pressure. Also, the resource consumption monitors gradual increase with the volume of attack. The system is less scalable when attacked. Upon completion of pass 3, Usnort is implemented. Significant improvement on resource availability can be seen along with some improvement over scalability.

VI. Conclusion and Future Work

The focus of the study was to investigate DDoS in detail and come up with an immediate solution to it. The objective of the paper is to propose a security tool to prevent DDoS attack on cloud environment. Short description of cloud systems, its features, advantages and deployment models have been presented. DDoS concept is explained and a security tool is proposed. Then, the tool is implemented in a distributed virtual environment. Two different scenarios have been given; When DDoS happens without implementing the proposed tool and second, when the proposed tool is implemented. CPU performance under both the conditions has been recorded and monitored. The result of the attack showed how it affects the cloud servers computational resources. It also illustrated that the use of Usnort can be very beneficial in terms of DDoS detection and prevention. Despite the fact that, some false alarms were generated for even a genuine user request, Usnort proved to detect DDoS on the network once it happens. It can be also seen with the experiment how Usnort can enhance the CPU performance under DDoS attack.

Cloud computing definitely has so much to offer in current computing paradigm. With increasing popularity, its challenges are also increasing. Security is the primary issue associated with it. DDoS is a serious security related attack that consumes all the CPU resources making it literally unavailable for further services. This paper diagnosed DDoS to some extent and proposed a tool based solution to it. Usnort showed acceptable performance to detect and prevent DDoS as shown by the experiments. However, due to various constraints, this paper could not address issues like quality of service, cost, and performance. Since tuning of one variable has equally significant impact on overall system, the overall behavior of the system could also be studied. Its effect over the cost needs a thorough cost benefit analysis for its effectiveness. The false alarm generated immediately after the attack hits the server could also be minimized. This sets out the future direction of research.

References

[1] K. Adamova, D. Schatzmann, P. Bernhard, and P. Smith, "Network Anomaly Detection in the cloud:The Challenges of Virtual Service Migration,", *2014 IEEE International Conference on Communications (ICC),* pp. 3770-3775,2014.

[2] A. DJENNA, and M. Batouche, "Security Problems in Cloud Infrastructures," *The 2014 International Symposium on Networks, Computers and Communications,* pp. 1-6,2014.

[3] R. Chalse, A. Katara, A. Selokar, and R. Talmale, "*Inter cloud data transfer security,"*.

[4] Arbor Networks, Inc. (2013). *Top Daily DDoS attack worldwide.* Retrieved 2015, from Digital Attack Map: http://www.digitalattackmap.com/understanding-ddos/

[5] F. Guenane, M. Noguiera, and G. Pujolle, "Reducing DDoS attacks impact using a Hybrid Cloud Based Firewalling Architecture," *Global Information Infrastructure and Networking Symposium (GIIS),* pp. 1-6, 2014.

[6] S.Maria, and S. Ahemad, "A QoS-oriented Inter-Cloud Federation Framework," *2014 IEEE 38th Annual International Computers, Software and Applications Conference* , pp. 642-643,2014.

[7] S. Dubey, S. Bhajia, and D. Tridevi, "Security Issues In Cloud Computing and Countermeasures," *International Journal of Innovative Science, Engineering & Technology* ,pp. 1-8,2014.

[8] S. Mohammad, A. Amirgholipour, S. M. Alirezanejad, B. Shakeri, and G. Mohammad, "Availability challenge of cloud system under DDOS attack," *Indian Journal of Science and Technology*,pp.6-10,2012.

[9] R. Rao, and P. Prakash, "Improving Security for data migration in cloud computing using randomized encryption technique," *IOSR Journal of Computer Engineering* ,vol.11, pp. 39-42,2013.

[10] A. Muhammad, A. and Eui-Nam-Huh, "Media Inter-Cloud Architecture and Storage," *2014 IEEE International Conference on Cloud and Autonomic Computing* , pp. 206-211,2014.

[11] https://www.snort.org/

[12] https://www.wireshark.org/

[13] M.Aiash, G. Mapp, and O. Gemikonakli, "Secure Live Virtual Machines Migration:Issues ans Solutions," *2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA),* pp. 160-165,2014.