Journal of Information Security, 2016, 7, 172-184
Published Online April 2016 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2016.73013

Generation of Labelled Datasets to Quantify the Impact of Security Threats to Cloud Data Centers

Sai Kiran Mukkavilli, Sachin Shetty, Liang Hong

Department of Electrical & Computer Engineering, Tennessee State University, Nashville, TN, USA
Email: smukkavi@my.tnstate.edu, sshetty@tnstate.edu, lhong@tnstate.edu

Received 18 February 2016; accepted 11 April 2016; published 14 April 2016

Abstract

Anomaly based approaches in network intrusion detection suffer from evaluation, comparison and deployment which originate from the scarcity of adequate publicly available network trace datasets. Also, publicly available datasets are either outdated or generated in a controlled environment. Due to the ubiquity of cloud computing environments in commercial and government internet services, there is a need to assess the impacts of network attacks in cloud data centers. To the best of our knowledge, there is no publicly available dataset which captures the normal and anomalous network traces in the interactions between cloud users and cloud data centers. In this paper, we present an experimental platform designed to represent a practical interaction between cloud users and cloud services and collect network traces resulting from this interaction to conduct anomaly detection. We use Amazon web services (AWS) platform for conducting our experiments.

Keywords

Amazon Web Services, Anomaly Detection, Cloud Computing, Datasets, Intrusion Detection, Network Traces

1. Introduction

Intrusion detection is a very interesting topic among the researchers. In particular, anomaly detection is of high interest since it helps in detecting many novel attacks. However, there has not been a proper application of this system in the real world due to the complexity of these systems, as these require continuous testing and evaluation and proper tuning prior to deployment [1]. The most ideal methodology for running these systems is to train them with real labeled network traces which consist of comprehensive set of intrusions and abnormal behavior.

Anomaly-based network intrusion detection systems (IDS) model patterns of normal activity and detect novel network attacks [2] [3]. However, these systems depend on the availability of normal profile pattern. But these patterns can change over a period of time due to various changes [2] [3]. This is a major challenge in itself as the availability of such datasets is very rare and the systems have to depend on one or more available datasets which lack understanding as they are heavily anonymized.

Another challenge is the comparison of IDS systems against one another. The lack of appropriate public dataset severely affects the evaluation of IDSs mainly affecting anomaly based detectors. Many existing datasets (KDD & DARPA etc.) [4]-[6] are static making them obsolete, unmodifiable, and irreproducible, despite being used widely. As with any other emerging internet technology, security is a major challenge for clouds especially for the migrating organizational data. These security risks can be well understood if we have access to the network traces in the cloud. To the best of our knowledge; there is no publicly available dataset which captures the normal and anomalous network traces in the interactions between cloud users and cloud data centers. Due to the ubiquity of cloud computing environments in commercial and government internet services; there is a need to assess the impacts of network attacks in cloud data centers. A systematic approach has been devised to design and develop an experimental platform designed to represent a practical interaction between cloud users and cloud services and collect network traffic traces resulting from this interaction to conduct anomaly detection. These network traces from the cloud are readily sharable and can be interchanged among collaborators and researchers without major privacy issues. This work has been geared towards reaching the aforementioned vision. This paper is organized as follows: Section 2 provides an insight into the related work; Section 3 emphasizes on cloud datacenter and its services; Section 4 & 5 talk about the execution and collection of normal and attack traces in detail; Section 6 emphasizes on ethics when performing experiments. And the paper concludes in Section 7 with details on future work and improvements to the dataset.

2. Related Work

Cloud security issues have recently gained traction in the research community where the focus has primarily been on protecting servers on cloud providers (securing the low level operating systems or virtual machine implementations). Unsecured cloud servers have been proven to be crippled with novel denial-of-service attacks. Most existing work on network traffic generation has not focused on applicability in the area of network security and evaluation of anomaly based techniques. The authors in Sommer and Paxson [7] have made observations on anomaly based network intrusion detection mechanisms and have provided recommendations to further improve research in this field [7]. They indicate that in order to improve the intrusion detection systems, datasets play a crucial role to know the system behavior. They also acknowledge that to obtain these datasets is very difficult and to do so it must be done with some collaboration with network operators. We have tried to implement the same in our work.

DHS Predict is a distributed repository of many hosts and providers at major universities and other institutions. Datasets mainly include Domain Name System (DNS) data, Internet Traffic Flow, Border Gateway Protocol (BGP), Internet Topology Data, Intrusion Detection System (IDS) and Firewall Data, and Botnet Behavior. Access to this dataset is available to certain verified accounts at some locations. Despite the major contributions by DARPA (Lincoln laboratory) [6] and KDD (UC Irvine) [4] datasets, they have not been able to reproduce the real world scenarios which is criticized in McHugh (2000) [8] and Brown *et al*. (2009) [9]. All these datasets are static making them obsolete, unmodifiable, and irreproducible, despite being used widely. Also the authors of the ISCX (2011) [10] dataset suggest a dynamic approach for generating the dataset, but this does not reflect the real world scenarios as the target servers they use are within the lab under the human assistance. Also, not much research has been done on the implication of vulnerabilities on the datacenter connecting the cloud user. In order to do so, datasets play a key role in demonstrating how well a system behaves. To the best of our knowledge, there is no publicly available dataset which captures the normal and anomalous network traces in the interactions between cloud users and cloud data centers. The systematic approach in this work addresses the flaws in the ISCX [10] dataset for generating a dataset dynamically and also shows the need for addressing the security issues in the cloud.

3. Overview of Cloud Data Center & Datacenter Services

Cloud computing is a general term for anything that involves delivering hosted services over the Internet. It is a nascent technology. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Supplying all those services at that scale requires can be achieved by expanding the hardware which makes up the datacenter. These services are delivered in various means (like private, public or hybrid cloud) by the cloud service providers (CSP) [11]-[16]. Examples include Amazon, Google, and Microsoft as shown in **Table 1**. A datacenter is usually a house of computers, and other components such as storage and network systems with other environmental supplies, backup power supplies and other security devices. Cloud Data centers are generally very huge, almost the size of multiple football fields (like Microsoft Azure datacenter). And maintaining them is a big issue too which costs millions of dollars [17]. So many companies lease space in large coalition facilities. Also many cloud regions are actually comprised of two or more distinct data centers (such as AWS in Sydney) [17].

3.1. Amazon Datacenters

AWS is located in 9 geographical regions : US East (Northern Virginia), US West (Northern California), US West (Oregon), AWS GovCloud (US) Region, Sao Paulo (Brazil), Ireland, Singapore, Tokyo and Sydney [17] as shown in **Figure 1** [18]. There is a dedicated GovCloud region located at Oregon, USA for US Government customers. All the data and the services stay within a designated region. When the user launches an instance the user can select an Availability zone or it is provided to him by Amazon. In order to prevent power outages within the zones they are isolated from each other. Our rented instances are located in the US-East (Northern Virginia) Region. We have chosen Amazon over other cloud providers for several reasons since Amazon's carriers (routers) security risks are lower compared to other carriers like Microsoft Azure [19]. Also Amazon has better pricing scheme (le carte pricing), where the users pay for what they use. There is couple of other benefits such as deployment speed, flexibility & performance.

3.2. Amazon Web Services (AWS)

Amazon web services (AWS) is an evolving and comprehensive cloud computing platform provided by Amazon.com. The first AWS was launched in 2006 to provide online services for websites. Sometimes web services are also known as remote or cloud services. AWS is distributed geographically into regions to ensure robustness and minimize the outages impact. The AWS offers many services like cloud drive, cloud search etc. [18]. These regions have central hubs located at Eastern USA, Western USA (two locations) Ireland, Australia, Singapore, Japan, and Brazil. Each region is divided into availability zones. Users of Amazon Web Services range from individual users (like students, professors etc.) to University research groups and small startup companies to large corporate businesses like Dropbox, Netflix. Etc. One of the main users of Amazon web service is the popular

Table 1. Table showing cloud datacenter locations of Amazon, Google, Microsoft.

PROVIDER	REGION & SUBREGION
AWS US	US East (N Virginia)
AWS US	US West (N California)
AWS US	US West (Oregon)
AWS	GovCloud (Oregon)
AWS	South America (Sao Paulo)
AWS	EU (Ireland)
AWS	Asia Pacific (Singapore)
AWS	Asia Pacific (Tokyo)
AWS	Asia Pacific (Sydney)
Google	Central US (Council Bluffs, IA)
Google	Central US (Pryor Creek, OK)
Google	Europe (Europe)
Microsoft	Azure North-central US (Chicago, IL)
Microsoft	Azure South-central US (San Antonio, TX)
Microsoft	Azure West US (California)
Microsoft	Azure East US (Boydton, Virginia)
Microsoft	Azure East Asia (Hong Kong, China)
Microsoft	Azure South East Asia (Singapore)
Microsoft	Azure Northern Europe (Dublin, Ireland)
Microsoft	Azure West Europe (Amsterdam, Netherlands)

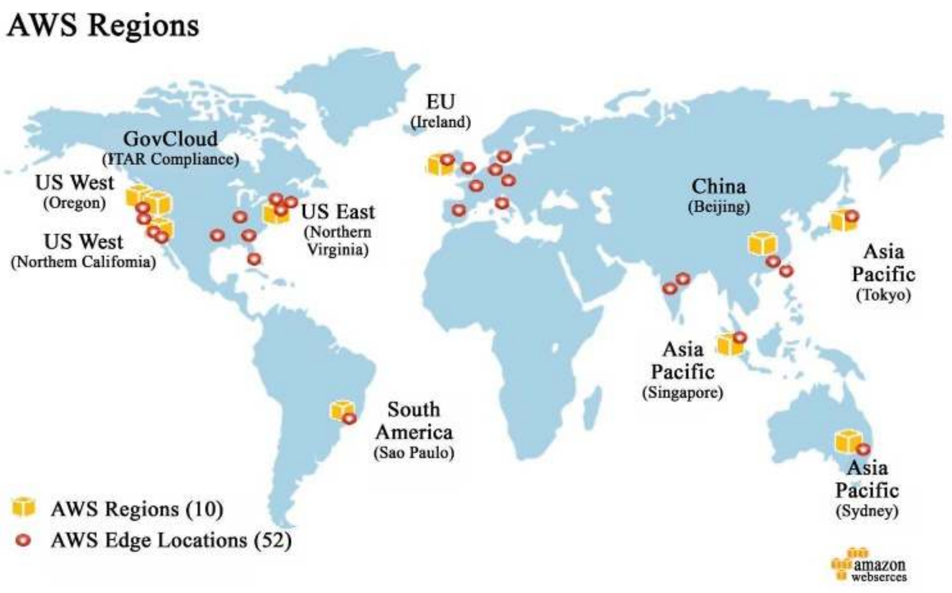

Figure 1. Map showing the location of Amazon cloud datacenters.

online storage service Dropbox which uses the IAAS (Infrastructure as a service) of the AWS. Once a file is added to Dropbox the file is transferred to Amazon S3 after encryption to various datacenters across USA. Similarly the download process is also the same. All AWS offerings are billed according to usage from service to service.

3.3. Cloud Users

Users access cloud computing using networked client devices, such as smartphones, desktop computers, laptops, tablets. The users are classified into two categories: Mobile cloud user's & Stationary cloud users. Mobile cloud users are the clients with access to mobile devices like smartphone, tablet etc. which use the resources of the cloud provider. Stationary users are the ones like desktop computers for accessing the cloud and also for performing research related to it. There are two main examples of stationary cloud users that are used for the research: PlanetLab, EmuLab. For our experiments we use PlanetLab nodes which mimic stationary cloud users. PlanetLab is a group of computers available as a test bed for computer networking and distributed systems research. PlanetLab is a great tool for performing large-scale Internet studies. Its power lies in that it runs over the common routes of the Internet and spans nodes across the world, making it far more realistic than a simulation. PlanetLab nodes utilize virtualization software, allowing applications to have full access to the system kernel [20].

3.4. Network Traffic Datasets

During the last decade, anomaly detection has attracted the attention of many researchers to overcome the weakness of signature-based IDSs in detecting novel attacks. There are often limitations to test and evaluate a novel network concept/solution on a real network. Hence, most researchers rely on captured network traffic data to evaluate the performance of their proposed network concept/solution. Also there is a scarce commodity among network research community for a real network traffic dataset. Various network problems have been analyzed, evaluated & validated based on captured network traffic. Hence it is important to maintain the completeness and quality of the network traffic dataset. There are many examples of network datasets like KDD CUP-99 [4], DARPA [6], and LBNL [21] etc.

3.5. Signature Based IDS

Intrusion detection systems (IDS) are classified into Anomaly based & Signature based. Signature based detection involves searching the traces (packets & bytes) for malicious traffic. The advantage of this technique is that if you know the network behavior you are trying to identify it is easy to develop and understand the signatures.

4. Generation of Normal Cloud Traces

One of the highest priorities of this work is to generate realistic background traffic. The main concern is to accurately reproduce the quantity and time distribution of flows for HTTP protocol (since majority of the traffic using the web is based on HTTP). To achieve this we have generated series of time instances which send web requests from the planet lab nodes to the EC2 server as shown in **Figure 2**. **Table 2** shows the nodes that are used to generate the requests. Each node has JDK installed in it which runs the java script for the requests.

To model HTTP requests, several approaches are available. The majority of work in literature is based on well-known statistical distributions. These probability distributions are analytically well described and have the advantage of being compact and easy to evaluate. We have generated series of random time instances using mean (μ) and standard deviation (σ) which follow both normal and uniform distributions separately. These are used to generate series of web requests from the PlanetLab nodes to the EC2 server. **Algorithm 1** explains in detail the mechanism for generating the web requests from normal nodes.

The algorithm (**Algorithm 1**) is executed in the PlanetLab nodes which are used as the clients for generating the web requests. For our experiments we chose 4 nodes for generating normal instances. Initially the start time (t{1}) of the node is calculated and then a time sequence (t) is read from the file. Then the thread is made to sleep for some time and then again the current system time (t{2}) is calculated. The time difference (d) between current time (t{2}) and start time (t{1}) is calculated. Then the difference (d) is compared with the time sequence (t). If (d) is more than (t), this is the time to start the web request (t{3}). The node generates a request to the EC2 server to which the server responds. There is a TCP Handshake taking place between the node and the server. Then the file is downloaded onto the node. The end time (t{4}) is calculated after the web request response is completed. The web response time (WRT) is the difference between the end time (t{4}) and start time (t{3}). In this way WRT is calculated for one sequence. If (d) is less than (t) then the new time sequence is read from the file. **Figure 2** shows the time instances generated using the normal distribution. These time instances are generated with a mean of 2.0216 and standard deviation of 0.3423. Similarly we generate time instances based on uniform distribution with a mean of 3.9209 and Standard deviation of 2.3446. The entire traffic is captured using the WIRESHARK running on the EC2 server.

5. Generation of Attack Cloud Traces

Since the proposed dataset is intended for network security and intrusion detection purposes [8], it would not be

Table 2. PlanetLab nodes for generating normal traces.

PlanetLab Nodes IP Address
pl2.eecs.utk.edu 160.36.57.173 pli1-pa-6.hpl.hp.com204.123.28.57 planetlab2.unl.edu129.93.229.139 planetlab2.cesnet.cz195.113.161.83

Algorithm 1: HTTP wget request logic

```
1:  begin:
2:  Get current system time $t1$
3:  t ⟵ Get time instance from the file
4:  thread.sleep()
5: Get current system time $t2$
6:  while t.hasnext() do
7:       index  $i \longleftarrow 0$
8:       diff of time $d \longleftarrow t2-t1$
9:       if $d>t$ then
10:           time $t3 \longleftarrow$ start the wed request
11:           wget file from EC2
12:           time $t4 \longleftarrow$ end  the web  request
13:           Web Response time d ⟵ $t4-t3$
14:      else
15:           thread.sleep()
16:      end if
17:  end while
```

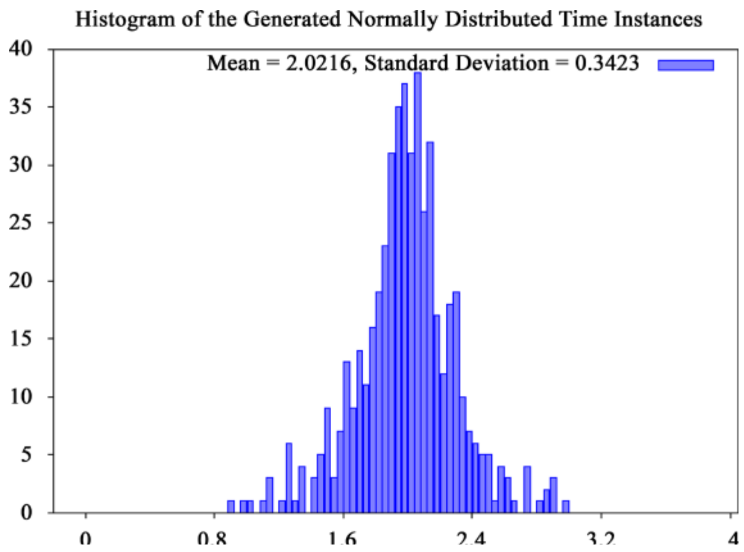

Figure 2. Time Instances based on normal distribution.

complete without a diverse set of attack scenarios. Attack traffic represents an attack scenario in an unambiguous manner. In the simplest case humans can carry out these attacks, and in the ideal case the autonomous agents can be used along with the compilers to carry out these attacks. Today, cloud computing systems are providing a wide variety of services and interfaces to the customers. There are various threats to these services which are explained in this paper. Our aim here is to mimic the actions of malicious hackers by performing multi-stage attack scenarios, each carefully crafted toward achieving a predefined set of goals [1]. The following are the common attacks that take place in the network. Denial-of-Service attack, Man-in-the-Middle Attack, Sniffer Attack, Portscan Application-Layer Attack etc. Out of these attacks we use the following three for the attacks in the cloud and for capturing traffic since these attacks deal with the Application layer protocols in the cloud and these best describe them and in capturing traffic.

- DDoS (Distributed denial of service)
- Man-in-the-Middle attack or ARP spoof
- Portscan

5.1. DDoS in Cloud

Distributed denial of service (DDoS) is an attack which many nodes systems attack one node all at the same time with a flood of messages. A distributed denial-of-service (DDoS) attack is one in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the targeted system [22] [23]. There are two types of DDoS attacks: a network-centric attack which overloads a service by using up bandwidth and an application-layer attack which overloads a service or database with application calls. For our experiments we use H-DOS in which we exploit seemingly-legitimate HTTP GET or POST requests to attack a web server. On July 17th 2013 a Distributed Denial-of-Service attack crippled the servers at hosting services firm Network Solutions, disrupting thousands of websites for several hours. DDoS attackers overwhelm servers by flooding a company's pipeline with unwanted network packets. Herndon, Va.-based Network Solutions, which manages more than 6 million domains, said on Facebook that its network security team was forced to respond to the attack. The outage is one of at least a dozen outages at cloud hosting providers impacting users in 2013. DDoS attacks are a common occurrence at hosting providers, e-commerce businesses and financial institutions [24]. In June, Network Solutions had its DNS servers hijacked and reconfigured to a malicious website after it botched efforts to thwart a DDoS attack.

5.2. Testbed Network Architecture for DDoS

The testbed network architecture for DDoS as shown in **Figure 3** consists of 8 PlanetLab nodes which are distributed globally but are interconnected. They are loaded with fedora operating systems. Out of these few nodes are used to launch the attacks and few to generate normal traces. To perform the experiments we have rented an Amazon EC2 instance with Windows server 2008 as the operating system. Amazon provides various instances which vary based on price, performance etc. There are various kinds like General, Compute, Memory, Storage & GPU. Out of this we have selected the General category. In the general category we chose t1.micro for our experiments. T1 Micro instances (t1.micro) provide a small amount of consistent CPU resources and allow you to increase CPU capacity in short burst when additional cycles are available. They are well suited for lower throughput applications and websites that require additional compute cycles periodically [18]. The rented instance is located in US-east region. The instance has the following configuration: Processor: Intel(R) Xeon(R) CPU E5430 @ 2.66 GHz RAM: 595 MB Cache: 6MB Address sizes: 38 bits physical, 48 bits virtual. Operating System: Windows Server 2008. The instance was launched and then an Apache Server 2.0 was setup to host our website. The website has a public IP of 72.44.46.206. We install and launch WIRESHARK in the same instance for capturing and storing the network traces. These traces are then moved to another system for monitoring and intrusion detection. The PlanetLab nodes used for the attack are shown in the **Table 3** and for generating normal traces are shown in **Table 2**.

5.3. Security Groups in EC2

When launching an Amazon EC2 instance we need to specify its security group. The security group acts as a firewall allowing us to choose which protocols and ports are open to computers over the internet. We can choose to use the default security group and then customize it, or can create our own security group. Configuring a security group can be done with code or using the Amazon EC2 management console [25] [26]. The default security group allows all the incoming traffic and leaves most of the ports open. In order to make the experiment more realistic we have customized the security group in the following way. We have allowed the following protocols from the following ports as shown in **Table 4**.

SSH was used so that remote hosts could communicate with the EC2 server, HTTP was used for the website to be accessible; RDP was used so that the server could be launched from our system, DNS so that the server could be accessed with a DNS. All the other protocols were blocked.

5.4. Load Balancing & Round-Robin DNS in EC2

Load balancing is when the processes and communications are distributed evenly across a network. When it's

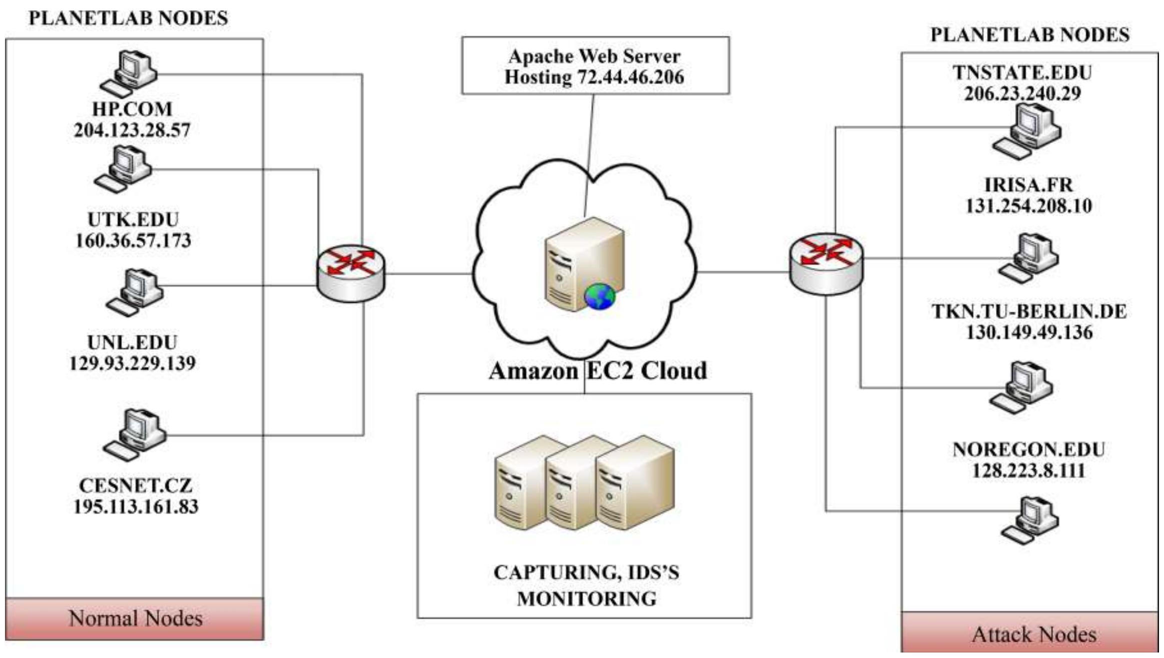

Figure 3. Testbed network architecture for DDoS [5.2].

Table 3. PlanetLab nodes for generating attack traces.

PlanetLab Nodes IP Address
peeramide.irisa.fr 131.254.208.10
planetlab01.tkn.tu-berlin.de 130.149.49.136
planetlab2.tsuniv.edu 206.23.240.29
planetlab1.cs.uoregon.edu 195.113.161.83

Table 4. Protocols allowed to communicate through EC2.

Protocols	Port
SSH	22
HTTP	80
RDP	3389
DNS	53

difficult to predict the number of requests that will be issued to a server we use load balancing. Busy Web sites typically employ more than one web server in a load balancing scheme. If one server is full of requests, the requests are forwarded to another server with more capacity [27]. We use a popular tool available online 'lbd.sh' [28] to determine whether the web server is load balanced or not. Lbd (load balancing detector) detects if a given domain uses DNS and/or HTTP Load-Balancing (via Server: and Date: headers and differences between server answers). After running the script on EC2, it was shown that EC2 does load balancing. Also we have tested whether the EC2 uses the Round robin DNS scheme, wherein the server has one domain name but multiple IP addresses. After testing it was found that the EC2 has only one IP linked to one domain name.

5.5. Implementing DDOS on EC2

This attack is designed toward performing a stealthy, low bandwidth distributed denial of service attack without flooding the network. We will be using "slowloris" [29] as the main tool in this scenario as it has proven to make web servers completely inaccessible using a single machine. The slowloris starts by making a full TCP connection to the remote server. The connection is held open by the tool by sending valid & incomplete requests to the server at the regular intervals to keep the socket from closing. Since the web server capacity is limited, in a certain amount of time all the sockets are used up and no other connection is made. We start the attack by deploying the "slowloris" script in the attack nodes. The attack is planned in such a way that all the machines start the attack within the same time window with a minimum lag. The attack nodes start running the script with each script sending 110 numbers of requests and with a "tcpto" 5 to port 80 of the EC2 server. The attack takes place at random intervals while the normal behavior (traffic) keeps on running continuously. The attack is a stealthy one since the magnitude and the frequency of requests are made to look similar to the normal behavior.

The attack slowly overwhelms the server thereby bringing down the service completely. When the attack stops the server starts automatically again. **Figure 4** & **Figure 5** show the attack and normal behavior of the network traces following different distributions. The traffic is captured using the wireshark on the EC2 and then is moved to another system for monitoring and IDS (Intrusion detection system).

5.6. ARP Spoofing in EC2

ARP spoofing is a technique where spoofed messages are sent by the attacker into the LAN (Local area network). The attacker machine sits anonymized in between the host and the gateway and captures the traffic both ways. The technique it uses for capturing the traffic is IP forwarding. Many of today's networks are built on what is called the eggshell principle: hard on outside and soft on the inside. This means that if an attacker gains access to a host on the inside, she can then use the compromised host as a pivot to attack systems not previously accessible via the Internet such as a local intranet server or a domain controller. In our case we host two machines in the same virtual private cloud (VPC) in Amazon EC2, one machine acts as host and the second machine will be the attacker. The attacker machine will capture the traffic between the host and the gateway as shown in the **Figure 6**.

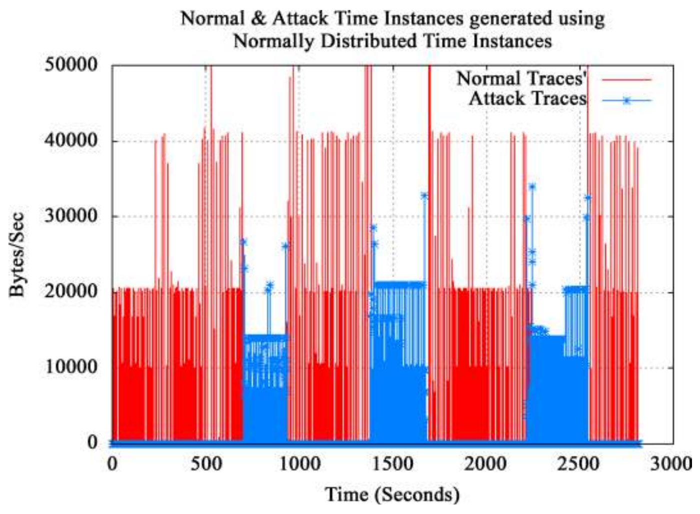

Figure 4. Network traces generated using normal time instances.

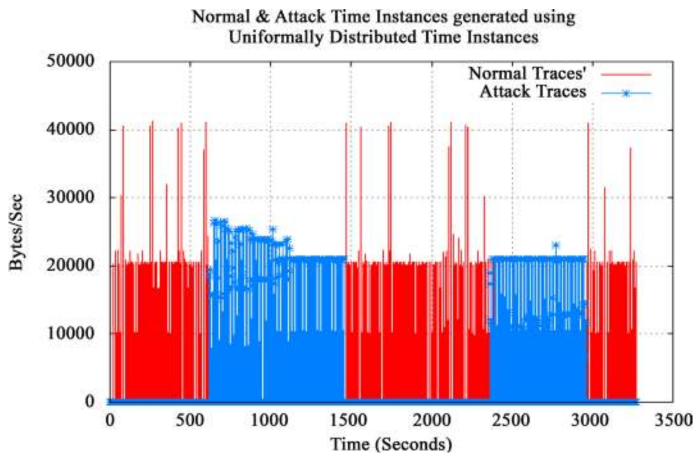

Figure 5. Network traces generated using uniform time instances.

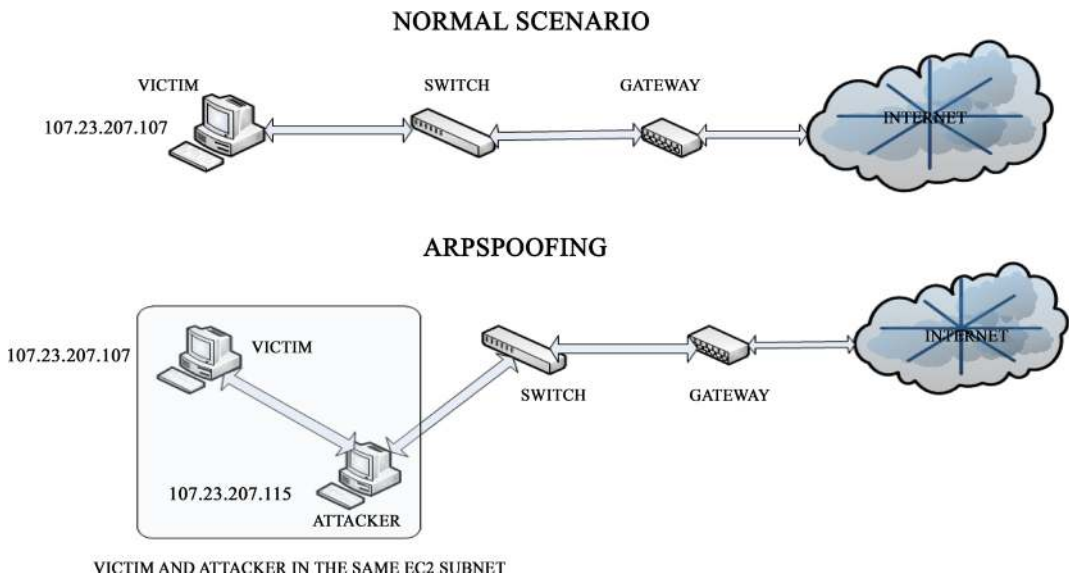

Figure 6. ARP spoofing in EC2.

For this experiment we consider the following: Two Windows 2008 server EC2 instances in the same subnet, Wire-shark, Ettercap-NG. We rent two instances with following configuration as shown in **Table 5**. The two instances are selected such that both are in the same subnet. For this experiment we first collect the traffic normally (in the absence of attacker) using wireshark. Then the well-known Spoofing software ETTERCAP-NG is installed in the attacker machine which listens to all the traffic between the victim machine and the gateway. Ettercap works by putting the network interface into promiscuous mode and by ARP poisoning the target machines. Thereby it can act as a "man in the middle" and unleash various attacks on the victims. For our experiment we have visited few websites like Facebook, Gmail etc. when the attacker is not present and again revisited the same websites in the presence of attacker. After collecting the traffic in both the attack and normal scenarios we convert it into user readable format (.arff) using tshark and then select the feature which best differentiates the normal and attack traffic. We have sorted the following feature from the list of features available "**tcp.analysis.ack_rtt**". This feature represents the round trip time. **Figure 7** shows the difference between attack & normal traces using the above feature.

5.7. Port Scanning in EC2

Port scanning is a technique where the open ports of a server or website are probed. It is used by attackers as a means to compromise the services running on a system. We use Nmap which helps us in providing the open ports and the services running on the server. For our experiment we use Nmap to detect open ports and also any other operating system vulnerabilities by launching stealth attack as shown in **Figure 8**.

Table 5. Description of rented EC2 instances for ARP spoof attack.

Operating System	IP Address	Type
Windows Server 2008	IP:107.23.207.107	Victim
Windows Server 2008	IP:107.23.207.115	Attacker

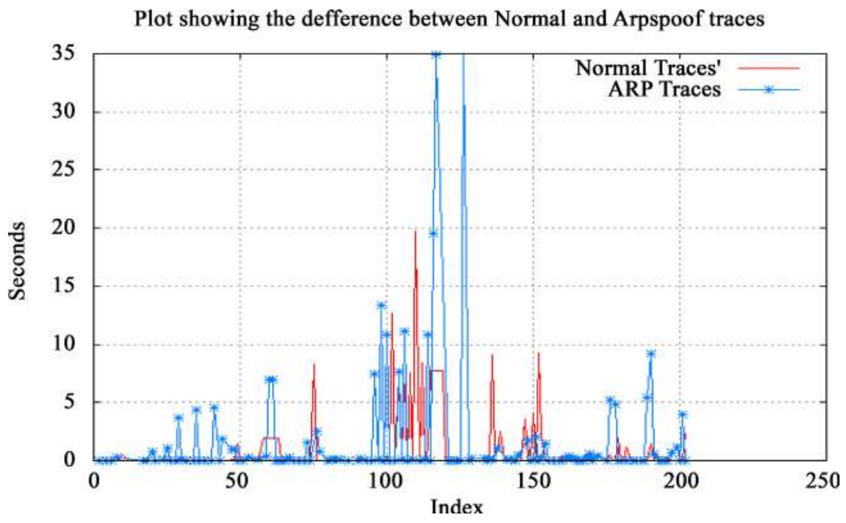

Figure 7. Difference between normal and ARP spoof traffic from same PlanetLab nodes.

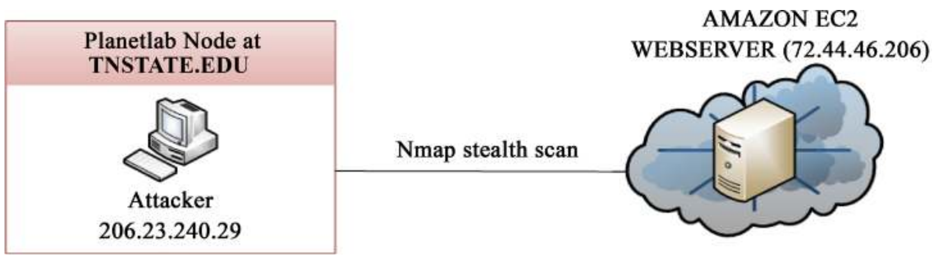

Figure 8. Portscanning using Nmap.

```
root@saikiran:~#sudo nmap -v -O --osscan-guess 72.44.46.206
Starting Nmap 4.52 (http://insecure.org) at 2014-11-01 18:10 UTC
Initiating Ping Scan at 18:10
Scanning 72.44.46.206 [2 ports]
Completed Ping Scan at 18:10, 0.07 s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 18:10
Completed Parallel DNS resolution of 1 host. at 18:10, 0.08 s elapsed
Initiating SYN Stealth Scan at 18:10
Scanning ec2-72-44-46-206.compute-1.amazonaws.com (72.44.46.206) [1714 ports]
Discovered open port 3389/tcp on 72.44.46.206
Discovered open port 80/tcp on 72.44.46.206
Completed SYN Stealth Scan at 18:10, 34.46 s elapsed (1714 total ports)
Initiating OS detection (try #1) against ec2-72-44-46-206.compute-1.amazonaws.com (72.44.46.206)
Retrying OS detection (try #2) against ec2-72-44-46-206.compute-1.amazonaws.com (72.44.46.206)
Host ec2-72-44-46-206.compute-1.amazonaws.com (72.44.46.206) appears to be up ... good.
Interesting ports on ec2-72-44-46-206.compute-1.amazonaws.com (72.44.46.206):
Not shown: 1700 filtered ports
PORT STATE SERVICE
80/tcp open http
3389/tcp open ms-term-serv
6000/tcp closed X11
6001/tcp closed X11:1
6002/tcp closed X11:2
6003/tcp closed X11:3
6004/tcp closed X11:4
6005/tcp closed X11:5
6006/tcp closed X11:6
6007/tcp closed X11:7
6008/tcp closed X11:8
6009/tcp closed X11:9
6017/tcp closed xmail-ctrl
6050/tcp closed arcserve
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows Vista|2008 (98%)
Aggressive OS guesses: Microsoft Windows Vista (98%), Microsoft Windows Server 2008 Beta 3 (96%),
Microsoft Windows Vista Home Basic (91%)
No exact OS matches for host (test conditions non-ideal).
Uptime: 61.417 days (since Mon Sep 1 08:10:23 2014)
TCP Sequence Prediction: Difficulty = 262 (Good luck!)
IP ID Sequence Generation: Incremental
OS detection performed.
Nmap done: 1 IP address (1 host up) scanned in 38.886 seconds
Raw packets sent: 5194 (232.752 KB) | Rcvd: 36 (2092 B)
```

For the purposes of proper scanning and pinging we had to allow ICMP traffic through EC2. We had allowed this rule in the EC2 security groups. The first session is initiated from the PlanetLab node. Nmap is launched at the "tsuniv.edu'' (206.23.240.29) PlanetLab node. It launches a stealth scan on the EC2 server which gives an approximation of the operating system and the number of open ports as shown above. The traffic is captured using the wireshark software [30].

6. Ethical Consideration

Our experiments to implement the framework for generation and collection of network traces involve real world instances and systems. This usually raises an ethical debate as scanning remote network devices can sometimes lead to adverse attacks. At the same time, developing a robust framework for network traces without collecting data from the real world is very difficult. Simulation tools and performing experiments with a controlled lab environment cannot replicate the randomness of the real world network traffic. A recent journal article that discusses the ethics of security vulnerability research [31], states that this type of zealous vulnerability research serves important social functions. Amazon EC2 provides students and researchers instances (Penetration Testing) that can be used for performing the experiments which involve attacks. For this the user has to get the permission from AWS before conducting any tests [32]. This approach is neither illegal nor unethical under the US laws. While accessing the instances to collect the vulnerability information we have taken utmost care not to disturb the host functions. We used minimum external resources to accurately collect the traces. The target networks in/24 blocks were scanned in a non-sequential order so that no organization is overwhelmed with our attacks. Also we did not scan any router or instance unnecessarily.

7. Conclusion & Future Work

Cloud computing offers many services to their clients including software, infrastructure etc., but they pose significant security risks to customer applications and data beyond what is expected using traditional on-premises architecture. These security risks can be well understood if we have access to the network traces in the cloud. Most of the network trace datasets are proprietary and cannot be shared due to privacy reasons; others are heavily anonymized and do not reflect current trends and lack certain statistical properties. Also, publicly available datasets are either outdated or generated in a controlled environment [1]. To the best of our knowledge; there is no publicly available dataset which captures the normal and anomalous network traces in the interactions between cloud users and cloud data centers.

Due to the ubiquity of cloud computing environments in commercial and government internet services, there is a need to assess the impacts of network attacks in cloud data centers. We present a systematic approach to design and develop an experimental platform designed to represent a practical interaction between cloud users and cloud services and collect network traffic traces resulting from this interaction to conduct anomaly detection. Our results show statistical differences between normal and anomalous network traffic traces which can be exploited by anomaly detection systems to detect and isolate adversaries in the cloud data centers. In future, we plan to implement the captured traffic on the IDS (Intrusion Detection Systems) for better understanding of anomalies and also to reduce the false positives.

Acknowledgements

This work was partially supported by Department of Homeland Security (DHS) SLA grant 2014-ST-062-000059 and Office of the Assistant Secretary of Defense for Research and Engineering (OASD(R&E)) under agreement number FAB750-15-2-0120.

References

[1] Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A. (2012) Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. *Computers & Security*, **31.3**, 357-374. http://dx.doi.org/10.1016/j.cose.2011.12.012

[2] Mukkavilli, S.K., Shetty, S. and Hong, L. (2012) Mining Concept Drifting Network Traffic in Cloud Computing Environments. *IEEE/ACM CCGRID*, Ottawa, 13-16 May 2012, 721-722.

[3] Shetty, S., Mukkavilli, S.K. and Keel, L.H. (2011) An Integrated Machine Learning and Control Theoretic Model for Mining Concept Drifting Data Streams. *IEEE HST*, Waltham, 15-17 November 2011, 75-80. http://dx.doi.org/10.1109/ths.2011.6107850

[4] University of California-KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html;2011

[5] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009) A Detailed Analysis of the KDD CUP 99 Data Set. *IEEE CISDA*, Ottawa, 8-10 July 2009, 1-6.

[6] MIT Lincoln Lab DARPA Data. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/index.html;2011

[7] Sommer, R. and Paxson, V. (2010) Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security & Privacy*, Oakland, 16-19 May 2010, 305-316.

[8] McHugh, J. (2000) Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection

System Evaluations as Performed by Lincoln Laboratory. *ACM Trans on Information System Security*, **3**, 264-294. http://dx.doi.org/10.1145/382912.382923

[9] Brown, C., Cowperthwaite, A., Hijazi, A. and Somayaji, A. (2009) Analysis of the 1999 DARPA/Lincoln Laboratory IDS Evaluation Data with Netadhict. *IEEE International Conference on Computational Intelligence for Security and Defense Applications*, Ottawa, 8-10 July 2009, 1-7.

[10] ISCX Datasets. http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html

[11] Cloud vs. Traditional Data Center. http://www.businessnewsdaily.com/4982-cloud-vs-data-center.html

[12] Classification of Data Center. http://www.datacenterknowledge.com/archives/2013/11/01/a-public-private-or-hybrid-cloud-debate-not-really/

[13] Data Center Types. http://research.gigaom.com/2012/10/4-types-of-data-centers/

[14] CAIDA Data Centers. http://www.caida.org/

[15] Cloud Platform. http://mindstormtools.com/2014/02/16/amazon-web-services-aws-and-the-new-google-cloud-platform/

[16] Cloud Intrusion. http://www.di.unipi.it/~hkholidy/projects/cidd/

[17] AWS Data Centers. http://www.turnkeylinux.org/blog/aws-datacenters

[18] Amazon AWS Instances. https://aws.amazon.com/ec2/instance-types/

[19] Reddy, S., Shetty, S. and Xiong, K. (2013) Security Risk Assessment of Cloud Carrier. 2013 13*th IEEE/ACM International Symposium on Cluster*, *Cloud and Grid Computing* (*CCGrid*), Delft, 13-16 May 2013, 442-449.

[20] PlanetLab Nodes. http://www.planet-lab.org/status

[21] LBNL-The Internet Traffic Archive. http://www.icir.org/enterprise-tracing/download.html

[22] Specht, S.M. and Lee, R.B. (2004) Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures. *Proceedings of* 17*th International Conference on Parallel and Distributed Computing Systems*, San Francisco, 15-17 September 2004, 543-550.

[23] Distributed Attack. http://searchsecurity.techtarget.com/definition/distributed-denial-of-service-attack

[24] DDOS Attack. http://www.crn.com/news/security/240158492/ddos-attack-behind-latest-network-solutions-outage.htm

[25] EC2 Instances. http://aws.amazon.com/ec2/

[26] EC2 Security Groups. http://blog.learningtree.com/understanding-amazon-ec2-security-groups-and-firewalls/

[27] Load Balancing. http://www.webopedia.com/TERM/L/load_balancing.html

[28] Load Balancing Tool. https://packetstormsecurity.com/files/46871/lbd-0.1.sh.txt.html

[29] Slowloris Tool. https://github.com/gkbrk/slowloris

[30] Wireshark Tool. https://www.wireshark.org/

[31] Benson, T., Akella, A. and Maltz, D.A. (2010) Network Traffic Characteristics of Data Centers in the Wild. *Proceedings of the* 10*th ACM SIGCOMM Conference on Internet Measurement*, Melbourne, 1-3 November 2010, 267-280. http://dx.doi.org/10.1145/1879141.1879175

[32] Penetration Testing Security. http://aws.amazon.com/security/penetration-testing/