Journal of Network and Computer Applications 34 (2011) 1097–1107

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks

Ashley Chonka, Yang Xiang*, Wanlei Zhou, Alessio Bonti

School of Information Technology, Deakin University, Australia

ARTICLE INFO

Article history:
Received 23 November 2009
Received in revised form
23 April 2010
Accepted 7 June 2010
Available online 23 June 2010

Keywords:
Network security
DDoS
XDoS
HDoS
Cloud computing
Traceback

ABSTRACT

Cloud computing is still in its infancy in regards to its software as services (SAS), web services, utility computing and platform as services (PAS). All of these have remained individualized systems that you still need to plug into, even though these systems are heading towards full integration. One of the most serious threats to cloud computing itself comes from HTTP Denial of Service or XML-Based Denial of Service attacks. These types of attacks are simple and easy to implement by the attacker, but to security experts they are twice as difficult to stop. In this paper, we recreate some of the current attacks that attackers may initiate as HTTP and XML. We also offer a solution to traceback through our Cloud TraceBack (CTB) to find the source of these attacks, and introduce the use of a back propagation neutral network, called Cloud Protector, which was trained to detect and filter such attack traffic. Our results show that we were able to detect and filter most of the attack messages and were able to identify the source of the attack within a short period of time.

1. Introduction

Today, cloud computing systems are providing a wide variety of services and interfaces to enable vendors to rent out spaces on their physical machines at an hourly rate for a tidy profit (Amazon EC2 2009; INetu, 2009; ElasticHosts, 2009). The services that are provided by these vendors can vary from dynamically virtual machines (Enomaly.com, 2009; Keahey et al., 2005; Nurmi et al., 2009; McNett et al., 2007) to flexible hosted software services (Laplante et al., 2008; Hewlett-Packard, 2009; Hibler et al., 2008; Lenk et al., 2009). Each machine and software shares the notion that delivered resources should be allocated and de-allocated on demand, at the same time as providing reasonable performance.

According to the recent e-crime study conducted in 2009 by the E-Crime Congress in partnership with KPMG, it found that online customers are most at risk and that risk increases as time goes by (KPMG, 2009). For example, the study reported that 63% of respondents said their customers were predominately affected by poisoned websites. The survey also reported that 40% of the total respondents said that there had been an increase in technical sophistication of these attacks against their customers.

With any new technology, there will be enthusiastic people who want to learn all about it so they can contribute to the wider community and others who want to exploit it so that they can gain some type of advantage. With the emergence of cloud computing, multi-billion dollar organisations like IBM, Amazon, Google and Ebay have already invested in cloud technology. If extortionists threaten to bring down their Cloud System with a Distributed Denial of Service (DDoS) attack, which is for this paper means many nodes systems attacking one node all at the same time with a flood of messages, it is usually better for a corporation to pay the ransom than see their systems go off line (Fowler, 2009). However, it is not only extortionists that can exploit cloud computing. For example, Amazon or Ebay competitors could also use known vulnerabilities to interrupt the normal operations of their cloud system so their customers move onto the next business that can provide them with the service they require. Renting out its sky-high computer infrastructure from Amazon, this actual example happened to the BitBucket.com cloud, who according to the report, went down for 19 h (Metz, 2009).

The variant forms of DDoS attack tools like Agobot (F-Secure, 2003; Sophos, 2009), Mstream (Dittrich, 2000) and Trinoo (Dittrich, 1999) are still used by attacker today. But most attackers are more inclined to use the less complicated web based attack tools like Extensible Markup Language(XML)-based Denial of Service (X-DoS) and Hypertext Transfer Protocol (HTTP)-based Denial of Service (H-DoS) attack due to their simple implementation and lack of any real defences against them (Chonka et al. (2008a)).

X-DoS and its distributed version, Distributed XML-based DoS (DX-DoS), described by Padmanabhuni et al. (2006) and demonstrated by Jensen et al. (2007), occurs when an XML message is sent to a Web Server or Web Service with malicious content to use

* Corresponding author. Tel.: +61 3 9251 7482; fax: +61 3 9244 6440.
E-mail addresses: ashley.chonka@deakin.edu.au (A. Chonka), yang@deakin.edu.au (Y. Xiang), wanlei@deakin.edu.au (W. Zhou), alessio.bonti@deakin.edu.au (A. Bonti).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.06.004

up all their resources. One example of an X-DoS attack is called a Coercive Parsing attack, which manipulates the Web Service Request when a Simple Object Access Protocol (SOAP) is parsed to it so that it can transform the content to make it accessible to applications. The Coercive Parsing attack uses a continuous sequence of open tags so that the CPU usage on an Axis2 web server becomes exhausted.

H-DoS attacks are discussed and implemented in a web article by Stewart (2007), which describes the attack as a HTTP Flooder that starts up 1500 threads so that it can send randomised HTTP requests to the victim web server to exhaust its communication channels. Stewart (2007), also points out there is no way to distinguish between legitimate and illegitimate HTTP requests and no way to filter such traffic.

In this paper we use our previous work on service-oriented traceback architecture (now called Cloud TraceBack) to defend against X-DoS attacks (Chonka et al., 2008a, 2008b, 2009) the area of cloud computing. We also cover in this paper the implementation of a previously devastating H-DoS attack that affected Iran. This ongoing cyber attack was coordinated by the Iranian opposition party that was successful at disrupting access to the pro-Ahmadinejad websites by using a 3 prong attack (Danchev, 2009a, 2009b). We use this attack as an example of bringing down a cloud system like Amazon EC2, and also use it to train our back propagation neural network called Cloud Protector (formerly known as X-Detector) to detect this form of attack and remove it from the system.

The contribution this paper makes to the field of cloud computing is that it is first to analyse how X-DoS/H-DoS attacks affect cloud computing using real attack traffic that is provided by the StuPot project (StuPot, 2009). The second contribution we make is by updating our previous Service-Oriented Traceback Architectural (SOTA) model to a Cloud model in order to focus on protecting cloud computing from X-DoS/H-DoS attacks.

The rest of the paper is made up of the following: Section 2 covers the related work done in the information technology field on security for cloud computing and the X-DoS/H-DoS attacks that threaten this security. Section 3 covers the Cloud TraceBack model and Chaos Protector. Section 4 covers the experiments and evaluations. Finally, Section 5 covers our conclusions and future work.

2. Related work

In this section we briefly cover some of the attacks that are currently used within cloud computing. We also cover previous research on SOTA, which is based on service-oriented architecture and service-oriented grid architecture. To conclude this section, we briefly cover the research done on X-DoS which is a DDoS attack that could affect cloud computing.

2.1. Cloud computing attacks

In the current research on cloud computing (Laplante et al., 2008; Lenk et al., 2009) most think of cloud computing as the virtualization of on-demand, elastic, scalable, resource that is service. But as Balding pointed out in his Rivest, Shamir, and Adelman (RSA) conference presentation, cloud computing is actually much more, and that it really is the abstraction of services (Balding, 2009).

Since cloud computing security follows the idea of cloud computing, there are two main areas that security experts look at securing in a cloud system: These are VM vulnerabilities and message integrity (Availability, Integrity and Confidentiality) between cloud systems. Some of the attacks that encompass both are: Rafal's Heap Overflow in I/O (CVE-2007-4496), Rafal's against Xen (CVE-2007-5497), Rafal's against Microsoft Virtual Server (CVE-2007-5497) and Greg McManus Shared Folders vulnerability in VMware (CVE-2007-1744).

2.2. Service-oriented traceback architecture

SOTA is a web security service application that is product-neutral (Chonka et al., 2008a, 2008b, 2009). Its main objective is to apply a SOA approach to traceback methodology. This is in order to identify a forged message identity, since one of the main objectives of X-DoS and DX-DoS is to hide the attacker's true identity. The basis of SOTA is founded upon the Deterministic Packet Marking (DPM) algorithm (Belenky and Ansari, 2003).

DPM marks the ID field and reserved flag within the IP header. As each incoming packet enters the edge ingress router it is marked. The marked packets will remain unchanged as they traverse the network. Outgoing packets are ignored. DPM methodology is applied to our SOTA framework, by placing the Service-Oriented Traceback Mark (SOTM) within web service messages. If any other web security services (WS-Security, for example) are already employed, SOTM would replace the 'token' that contains the client identification. Real source message identification is stored within SOTM, and placed inside the SOAP message. SOTM, as in DPM tag, will not change as it traverses through the network. The composition of SOTM is made up of one XML tag, so not to weigh down the message, It is then stored within a SOAP header. Upon discovery of an X-DoS or DX-DoS attack, SOTM can be used to identify the true source of forged messages.

SOTA does not directly eliminate an X-DoS or DX-DoS attack message. This is left for the filter section of a defence system called Cloud Protector. This leaves SOTA with the important task of dealing with the main objectives of X-DoS and DX-DoS, which are:

- Exploit a known vulnerability or to flood the system with useless messages to exhaust the web server's resources to the point of collapse. These vulnerabilities could be found in communication channels (flooding for example) or known exploits within the services provided (for example, an attacker can overload their messages, which will result in the web server crashing).
- Attackers who try to hide their identities. The reasons for this vary, depending on the type of attack, but usually it is to cover their crime or to bypass a known defence that is in place to prevent it. It is with this second objective that SOTA attempts to cover, as other traceback methods do, items like Probability Packet Marking (PPM) (Savage et al., 2001) and DPM.

There are a number of reasons why cloud computing should employ a SOTA type framework:

- Current web security is not up to handling an X-DoS or DX-DoS attack. In fact, Jensen et al. (2007), shows how WS-Security can be used in an X-DoS attack.
- With IPv6 coming into use (van Beignum, 2008), current IP traceback methods will no longer be viable. This is due to the changes IPv6 introduces, such as, IPSec and the packet header format which no longer holds support for the fields that are required for IP traceback.
- SOTA does not violate IP protocols due to storing information in the IP packet.
- Using the SOA model, SOTA can be employed on any ubiquitous grid system.

2.3. XML-based denial of service (X-DoS) attacks

A Denial of Service (DoS) is where an attacker attempts to deprive legitimate users of their resources (Rogers, 2009). An X-DoS

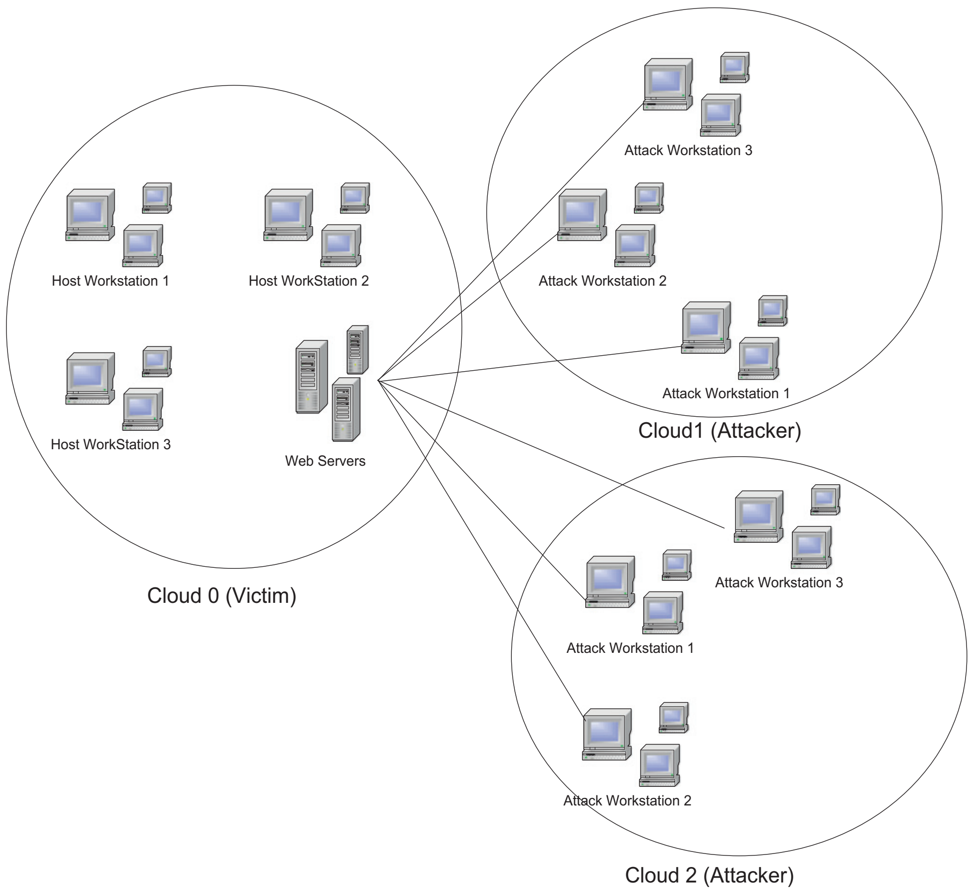


Fig. 1. Distributed XML-based Denial of Service attack, where an attacker has taken control of 2 cloud networks and starts to sending huge amounts XML-based message to Cloud 0 Web Server.

attack, according to Padmanabhuni et al. (2007) is where a network is flooded with XML messages instead of packets in order to prevent legitimate users to access network communications. Further, if the attacker floods the web server with XML requests, it will affect the availability of these web services. Attackers can also manipulate the message content, in order to cause the web server to crash as shown in Jensen et al. (2007).

To adapt X-DoS into a Distributed Denial of Service paradigm, called Distributed XML based Denial of Service (DX-DoS), the attacker uses multiple hosts to attack the victim with X-DoS attacks (Fig. 1). Though none of these attacks have been reported as yet, this type of attacks could be a very serious threat facing cloud computing in the future.

3. Cloud traceback for cloud computing

3.1. Introduction

One of the reasons why attackers are so successful is because of the internet characteristic, which comprise of limited and consumable resources. Attackers can target bandwidth, processing power and storages capacities of a cloud network. Cloud computing has limited resources so it has to provide a highly quality service, however these services can be exhausted with a sufficient number of consumers. With this particular knowledge, attackers can instigate an X-DoS or DX-DoS attack. For example, an attacker can open up a number of browsers in virtual machines so that it can send multiple requests to the victim's web server over a period time.

In a DX-DoS attack, the attacker would order their agents/zombies to instigate a flood attack of oversize messages, against the web server. This again would result in the web server crashing from either executing the oversize messages, or from communication congestion created from the flood.

Fig. 2 shows the placement of the cloud traceback defence system to protect the cloud system from X-DoS, DX-DoS and H-DoS attacks.

3.2. Cloud traceback description

Cloud TraceBack (CTB) can be used in either a network structure, such as a LAN, or a grid network structure. CTB is made within a virtual machine to make placement within the cloud network compatible, flexible and scalable whilst remains a SOA security product.

3.2.1. CTB placement within cloud system infrastructure

CTB is deployed at the edge routers in order to be close to the source end of the cloud network. Usually, if no security services are in place for web services, as seen in Fig. 1, the system becomes quite vulnerable to attacks. Fig. 2 demonstrates how CTB can remedy this by being located before the Web Server, in order to place a Cloud Traceback Mark (CTBM) tag within the CTB header. This is accomplished by attaching the Web Service Definition Language (WSDL) to the CTB instead of the web server.

As a result, all service requests are first sent to the CTB for marking, thereby effectively removing the service provider's address and preventing a direct attack. If an attack is discovered or was successful at bringing down the web server, the victim will

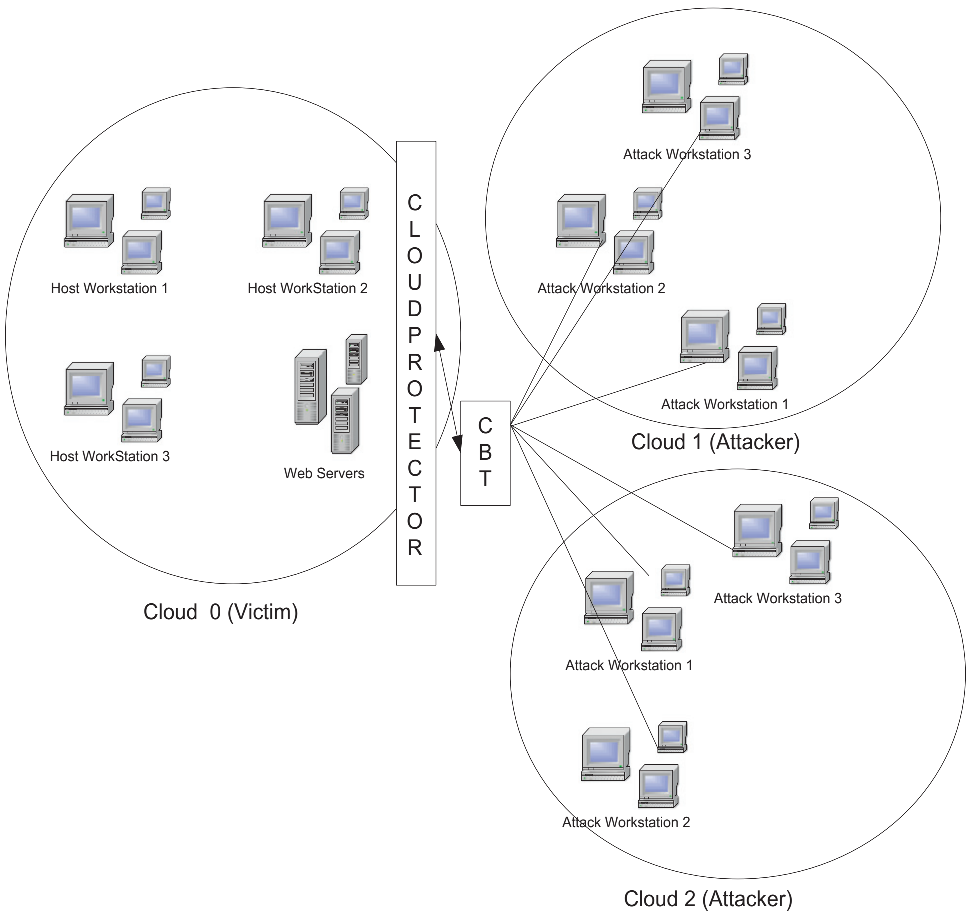


Fig. 2. Distributed XML-based Denial of Service attack, where CTB and Cloud Protector are located just between the each could Web Service, in order to detect and filter X-DoS attacks.

be able to recover and reconstruct the CTBM tag and as a result reveal the identity of the source.

In an attack scenario, the attack client will request a web service from CTB, which in turn will pass the request to the web server. The attack client will then formulate a SOAP request message based on the service description formulated by WSDL. Upon receipt of SOAP request message, SOTA will place a SOTM within the header. It is assumed that WS-Security would replace wsse username tag with its own username tag.

Once the CTBM has been placed, the SOAP message will be sent to the Web Server. Upon discovery of an attack, the victim will ask for reconstruction to extract the mark and inform them of the origin of the message. The reconstruction will also begin to filter out the attack traffic.

If the message is normal, the SOAP message is then forwarded to the request handler for processing. Upon receipt of the SOAP request, the Web Service will prepare a SOAP response. The web server then takes the SOAP response and sends it back to the client as part of the HTTP response. CTB will not interfere with the response requests or any outgoing message.

3.3. Cloud Protector

The Cloud Protector is a trained back propagation neural network (NN), to help detect and filter out X-DoS messages. A neural network is a set of connected units made up of input, hidden and output layers. Each of the connections in a neural network has a weight associated with it. In a neural net the focus is on the threshold logic unit (TLU). The TLU inserts input objects into an array of weighted quantities and sums them up to see if they are above the threshold.

3.4. Cloud traceback approach to SOA

Cloud TraceBack has a number of basic properties and characteristics of the service model (He et al., 2005). These characteristics are as follows (Ye and Singh, 2007):

- Loosely coupled: CTB is made from XML based language. This means that it can run on different platforms regardless of the programming language.
- Message based interaction: The interaction between the client, CTB, and service provider are all message based.
- Dynamic discovery: WSDL is attached to CTB so that all services are known to the public. This means that any client can connect to CTB at any time over the Internet and access the services of the service provider.
- Late binding: CTB and the service provider all run in real-time. This allows clients to access services whenever and wherever they are.
- Policy based behaviour: It is planned that a CTB-Policy will be developed in the future, following along the same guidelines as the WS-Security Policy. This policy will dictate messages marking procedures.

CTB acts like a service broker within a SOA model (see Fig. 3), which is a repository for service descriptions, such as WSDL or Universal Description, Discovery, and Integration (UDDI).

3.5. CTB's algebraic approach to determining path and reconstruction

3.5.1. Assumptions

The assumptions made about our first group of experiments are as follow:

- An attacker may control any number of client machines that can be widely distributed across the Internet.
- Attackers may know they are being traced.
- It only takes a few messages to get CTB reconstruction to begin traceback.
- CTB has not itself been compromised by the attackers.
- That the web service provider of web service has limited resources.
- SOAP headers are being used by the client.
- Real source ID is the location of the edge router.

3.5.2. Algebraic coding of path

The approach taken by CTB to encode the traceback information follows the same lines as Dean (2002), in which a polynomial is used within the tag so that the reconstruction of the path the message has taken can take place. With the use of a polynomial $f(x)$ of degree d in the prime field $GF(P)$, we can then use $f(x)$ given that $f(x)$ evaluated at $(d+1)$ unique points. Let $fP(x)=M_1,M_2,\ldots,M_n$.

3.5.3. Path reconstruction by CTB at cloud victim end

As shown in Fig. 4, when the message comes into the Cloud Victim (CV), CTB reconstruction will attempt to locate the source of where the message came from. Let k denote the number of attack paths the attacker might use and let L to denote the expected length of an attack path. Note: In this paper we assume that the attack paths are close to L in length.

Following Dean (2002) the complexity of path reconstruction by CV is $O(Lk^2)$ where the exponent of two reflects the brute force task that the victim must try in his/her attempt. Note: The reason why CV is not $O(Lk^8)$ as outlined in Dean (2002) is because with messages we are able to fit more information into a tag than an IP packet.

Another advantage of using messages is that a distance field can also be introduced via the use of the time-to-live value within the packet that the message comes in. With this distance field, it simplifies path reconstruction by allowing the CV to select a sample of messages for which its TTL is measured by $w\ dist=\ell$ for any given ℓ, therefore as long as the attacker is within the distance of ℓ and the routers have not been reset at distance ℓ then it can be assumed the messages are legitimate.

To know if the attacker has stepped within the distance of ℓ, we use the *Bleichenbacher–Nguyen* reconstruction algorithm. Let the values of $x_1,\ldots x_n$ be all the possible values of x. Each set S_i has the size at most of j, which is used to find all the polynomials a of degree at most d such that for all i. With each set of S_i it contains the distinct y value such that (x_i,y) occurs within the random messages that were sampled. The polynomial a is then set to the polynomials that correspond to the attack path which the attack message has taken.

In an example of how this is applied, the CV selects a sample of N messages to test and for each x_i chooses a set S_i of the size of m from all of the (x_i,y) points in the sample. If the $y>m$ in the (x_i,y) sample, then CV chooses which m values to include in S_i as attack messages from that particular attack path.

To recognise if the path has a 'false positive' (which is a polynomial that does not correspond the identification of the message and the reconstruction algorithm) in the random sample, it should be around.

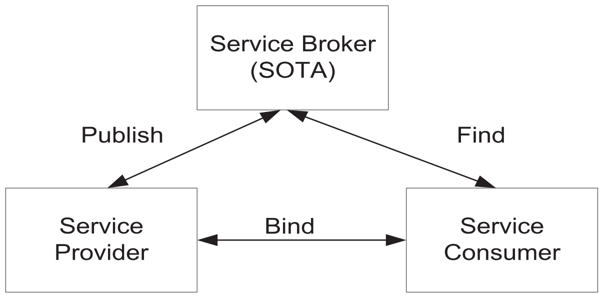


Fig. 3. SOA diagram with SOTA as the Service Broker.

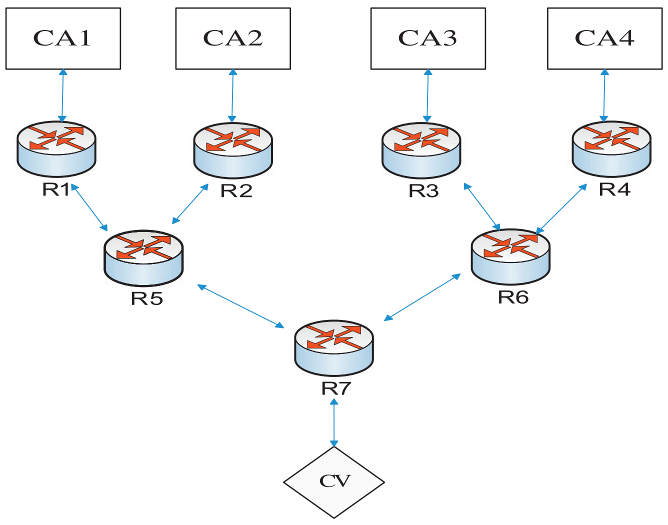


Fig. 4. True path from the cloud attackers to the cloud victim.

4. Evaluations

4.1. Introduction

In this section we conducted the following experiments and evaluations of a HTTP DoS attack, which were reproduced from the actual attack that hit Iran in June 2009 (Danchev, 2009a, 2009b). The purpose of this experiment is to see how effective an HTTP DoS attack is within a cloud computing environment.

The experiment and evaluation of CTB is discussed in a number of areas. The first area is the simulation of how a CTB works at marking incoming messages and to see if a message can be identified as X-DoS. The second area is how accurate CTB is at identifying the source of where the message comes from. We also want to know if CTB is a better performing security method when compared with SOAP authentication or WS-Security in regards to X-DoS attacks. Finally, we examine the results that were produced from the experiments to see what interpretation we can gather within area of cloud computing.

In addition to the areas discussed above, an important point is that even though CTB is a handy tool to identify the source of an attack within a cloud system that is running web services, it does not deal directly with removing the X-DoS threat or preventing X-DoS messages from causing problems. Therefore, with the experiment and evaluations within a cloud computing system the focus will also be on the Cloud Protector to see if it can detect and filter H-DoS and X-DoS messages.

4.2. HTTP DoS attack experiment and evaluation

4.2.1. Background on the DDoS attacks on Iran using web services

This section covers the DDoS attacks on Iran, with attackers using web-based attack tools like Page Rebooter, IFrames and Do-It-Yourself (DIY), which were used in a 3 pronged attack.

The Page Rebooter website was used for the first part of the attack (Pagerefresher 2009). The attackers inserted the name of the websites they wanted to attack, and with the use of a rebooter which allows to set the interval for refreshing a particular page, which in this case they were setting it at 1 second apart. Then, the attackers went on Twitter and various discussion forums announcing how to use the Page Rebooter website, by opening it up in many browser tabs and just leaving them to refresh the Iran Government website. According to Danchev (2009a) one attacker announced that he had set up 8 virtual machines running on 8 CPUs to run this variety of attack.

The second part of the 3 prong attack was to use an IFrames loading script, which automated the refreshing of the farsnews.com; irna.jr and rajanews.com websites. This resulted in the server crashing and the website page being a complete mess with half of the pictures and content missing from the page.

The last stage of the attack included a DIY, which is a denial of service tool that contains varies attack programs. These tools are BWRaeper.exe, PingFlooder.exe, Server_Attack_By-_C-4.exe and SupportIran.php. Each includes a simple manual and links to large images at the targeted websites.

4.2.2. Experiment

The experiment we conducted was to follow a Twitter post from an attacker who was an active participant at that time (Danchev, 2009a). By following his/her example, we opened up three virtual servers that contained 20 Firefox browsers and 20 open tabs to each browser (see Fig. 5). Note: The original attacker had 8 virtual machines (though how many browsers he/she had opened was not stated. We thought 20 browsers and tabs would be sufficient for demonstration) up at that time, but we thought 3 would be efficient to see the affects of the attack. With these open tabs, we then used our own version page refresher website that closely followed the PageRebooter (2009). Note: Fig. 5 displays the Page Rebooter websites for demonstration purposes only it was not used to generate our HTTP DoS attack.

We then set the targeted website to be refreshed, which was StuPot Project Web Server and then set the timer to 1 s. Tshark (Wireshark.com, 2009) and tcpdump (tcpdump.com, 2009) were used to collect the http traffic.

4.2.3. Evaluation of the attack

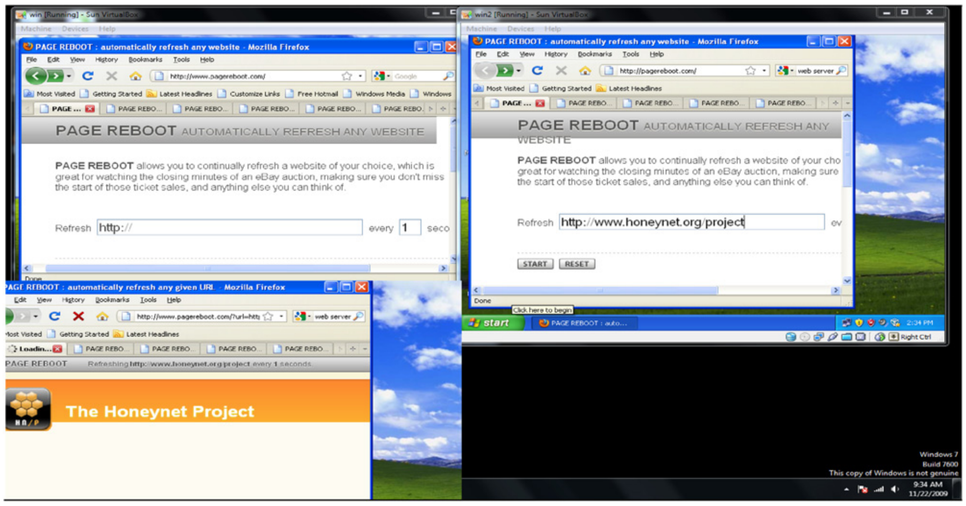
As we can see in Fig. 6, the attack started around 0:30 (point A). At this point in time, we started to see how the drain of http request was beginning to drain the virtual web server until it could only handle a few requests around 1:00 (point B). The web server then appears to recover within 5 min, probably due to O/S congestion controls, but as the attack continued, the web server again was reduced to handling only a few http requests at 1:10 (point C). We stopped the attack at around 1:15, but as can be seen with the spikes (point D) the attack had a lasting impact on the web server.

Though we do not have any data released by the Iranian Government on the attack, the Georgia Government (2008) did release a report on the HTTP attack that affected them.

If we have a look at Fig. 7 in a qualitative comparison with our data, we see some striking characteristics. Before the start of the attack, normal traffic flowed to point A, which saw the beginning of the attack. At point B, where there seemed to be a slight recovery the attack continued to increase the amount of http requests until it reached point C. This is where the web server congestion control probably started up or the system

```
Identification reconstruction procedure at web server
For each message request w from source Sx
        Create a table array
        Ws.tx = extract Transactioninfo()
        Ws.tx.time_and_date = timestamp
        Ws.tx.usernameId = usernameID
        Table_array[]+ = Ws.txusernameID
End
Display of username at particular time of the attack
For each Table_array[]
        Get what time of attack
        Get usernameID from Table_array[]
Display usernameID
```

Fig. 6. HTTP DoS attack hit around 0:30 at point A until the web server reached a peak of being able to handle only a few requests at a time around 1:00 points. The web server seemed to recover but with the continued http attack, the web server was still only able to handle a few http requests (traffic) at point C. Point D shows that after we stopped the attack the web server still slightly affected by the attack.


PAGE REBOOT : automatically refresh any website - Mozilla Firefox

PAGE REBOOT AUTOMATICALLY REFRESH ANY WEBSITE

PAGE REBOOT allows you to continually refresh a website of your choice, which is great for watching the closing minutes of an eBay auction, making sure you don't miss the start of those ticket sales, and anything else you can think of.

Refresh http:// every 1 seco

Refresh http://www.honeynet.org/project

PAGE REBOOT : automatically refresh any given URL - Mozilla Firefox

PAGE REBOOT Refreshing http://www.honeynet.org/project every 1 seconds

The Honeynet Project

Windows 7 Build 7600 This copy of Windows is not genuine

Fig. 5. Demonstration of our three virtual machines with 20 Firefox browsers and 20 tabs open to the Page Reboot Website. The purpose of our demonstration was to replicate the attack that brought down the Iranian website.

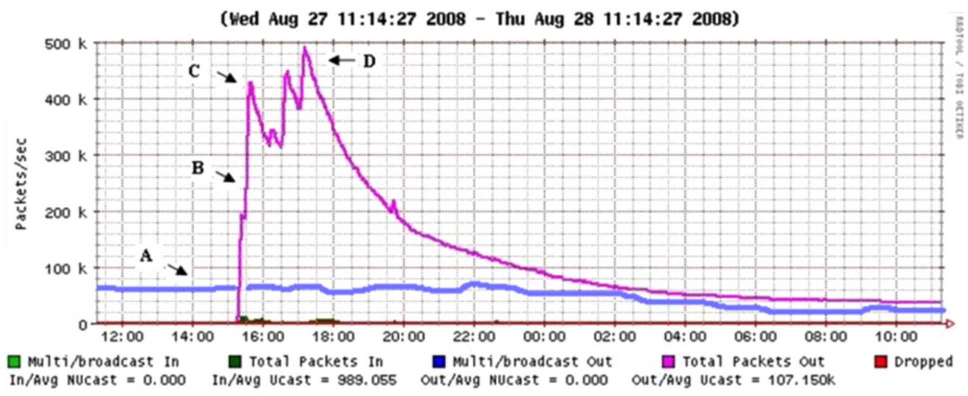


Fig. 7. The Georgian Ministry of Foreign Affairs web server's total packets out, showing the number of request's at the time of the attack. Point A has been extrapolated out from the end of attack to represent what the normal request traffic (self-similar) would have looked like if the attack had not taken place.

administration initiated some defence controls. But as we can see, they only prevailed for a small amount of time and the attack reached its peak at point D. The reason why the web server started to return to normal is not given in the report, but it is possible that it is due to a change of IP address.

As we can see from Fig. 7, a simple H-DoS attack is a very effective attack and to a cloud system, a very viable threat. Currently the only defence against it is to restrict the flow rate of HTTP requests coming into the server by using a proxy server or request the ISP to change the IP address. In the experiment and evaluation section of the Cloud Protector we will show how much of the attack data it is able to detect and filter.

4.3. Experiments using CTB

These experiments were performed on a Dell Dimension DM501 Intel Pentium single-core CPU, 3.0 GHz, 2 GB of RAM and 2 300GB SATA hard-drives. All the programs were implemented on a VM Server 2 (Vmware, 2009) with a Windows XP (Professional) ISO image. The software that was installed was .NET Web Services along with VB.Net Figs. 8 and 9 display the algorithms used to insert and extract the CTBM tags.

The first group of experiments were broken up into two sections: The first section of experiments simulated X-DoS attacks against a cloud service provider like Amazon EC2. This X-DoS attack follows the outline within (Jensen et al., 2007) by selecting the XML injection to replicate.

To simulate the success of an attack, the introduction of a random element of chance (50/50) that the message might crash the web-service was made. If the web server did not crash, the service provider was able to trace the message source and initiate filtering procedures. However, if the attack was successful, no more messages would be generated. Upon the web server's crash, it was assumed the service provider would restart it. Upon the restart, the service provider would access CTB reconstruction and find the source of the attack. Filtering of the X-DoS traffic is left to the experiments and evaluation section of the Cloud Protector.

Following on from the above, the second section of the experiments is done to compare CTB against SOAP authentication and WS-Security (Nadalin et al., 2008a, 2008b). The reason for this are the two well known service security applications to protect cloud services.

For the experiments, two things are needed, the header information and ID reconstruction. The header format for CTBM is changed after the SOAP message has been re-formatted.

Upon arrival, CTB looks at the SOAP header information on identification, and extracts from the message. If there is no SOAP header, a header is created with the client identification attached.

```
CTBM procedure at CTB, edge Interface I

For each incoming request message w
If no header then
   create SoapHeaderAttribute("client id")
   Invoke Header new SoapHeaderAttribute
Else
   get WSSusernameToken(xx)
   WSSusername = new client id
```

Fig. 8. Pseudocode to extract Header of the message that is coming into CTB.

```
Identification reconstruction procedure at web server
For each message request w from source Sx
        Create a table array
        Ws.tx = extract Transactioninfo()
        Ws.tx.time_and_date = timestamp
        Ws.tx.usernameId = usernameID
        Table_array[]+ = Ws.txusernameID
End
Display of username at particular time of the attack
For each Table_array[]
        Get what time of attack
        Get usernameID from Table_array[]
Display usernameID
```

Fig. 9. Pseudocode to extract, store and display username identification.

Once the header has been created or updated, the message is forwarded to the web server.

With ID reconstruction, CTB reconstruction is capable of handling the reconstruction of the path back to the true source of the message. In addition, CTB reconstruction is given the instruction by the service provider at the time of the attack or at the end to start the reconstruction process. The information that is extracted should lead to the source of the message attack and initiation of protection measures.

4.4. Evaluation of CTB

The simulation conducted was a message flood attack, using XML injection to replicate. The simulation program was set up to generate a total of 100 messages, and if one of these messages was an attack, then it had a 50/50 chance to crash the web server. If the system did crash, 50 ms was added to the next attacks response time. This response time would steadily decrease over time by 5 ms for every 5 messages that went by but did not crash the server. This was to simulate the time taken by the service provider to restart their system, locate the source, and filter it.

From the results in Fig. 10, 84 normal messages were sent over the network, with an unusually high rate of 9 successful attacks that crashed the system. It is surmised, that the reason for the crashes was due to the random chance that we built-in to our code, therefore if re-run the number of crashes would probably have been lower.

In addition to this, the results show 7 successful attacks were identified by SOTA (shown in Figs. 10 and 11). Fig. 11 shows the rate at which the CTB was able to identify the source of the 7 attacks, while points A, C and E show that there were 3 crashes (each giving a 50 ms increase to the response time).

Points A and B are interesting because there is a decrease of about 70 ms that indicates there was no successful crash even though there was one attack message between them that did not crash the web server. After point B though, it shows that it recorded the web server crashed two more times out of the total nine crashes recorded. It is interesting to note that at one stage we can see at points C and D there seem to be some recovery. This was probably due to the web server trying to compensate or right itself.

In summary, the above results in Figs. 10 and 11 show that the CTB over a cloud system can detect and identify the seven attack messages with a response time between 480 and 550 ms.

4.5. Evaluations of the second group of experiments

The second group of experiments deals with a basic SOAP Web Service being developed by using .Net and VB.Net. The program contained a basic header for authentication purposes. To simulate CTB, the program extracted the name id from the header and replaced with the real user id (010101). It is assumed that a one-way transmission delay between client and web server is 10 ms. The delay is simulated by the program going into a wait mode for 10 s, and this is added to the response time data. The measurements used in this experiment were the processing time over the response time.

The result shown in Fig. 12 took over 2 s of processing time, demonstrating CTB was far more effective than the SOAP authentication procedure. One of the reasons for the quicker response time was due to the CTB's swapping the tag and not going through a process such as authenticating the message itself. Therefore the extra response time leads to a reduction of computer resources during X-DoS attack.

Furthering the evaluation of Fig. 12, the results show at point A and B, CTB started to slightly increase, while SOAP authentication started to decrease, even at points C and D there seems to be a slight wave within the curve. More than likely, this is just the normal way that authentication and CTB behave. The results of Fig. 12 demonstrate that the CTB is far more effective security application than the SOAP authentication procedure when it comes to X-DoS attacks.

Another experiment was conducted in comparing CTB against WS-Security to see if the conclusion of this experiment can be verified in comparing CTB against SOAP authentication. The experiment was run with a WS-Security interaction application against Amazon EC2 (Amazon Web Services, 2009) along with the WS-Security application that contained a signed certificate for authentication purposes. For the purpose of this experiment, all we are looking to do is exchange the username ID for the authentication name, which was done before it was sent to the Amazon SOAP service. This was to ensure that the message would be received and that we would receive a response. The results are based on how long the application took to process a response from Amazon and SOTA was used in conjunction with WS-Security, in order to replace the name ID for the real-source ID.

The results in Fig. 13 show that by introducing CTB into WS-Security, an increase in response time was generated by thirty percent. This increase means that during a DDoS attack, more processing time is required to handle the extra burden. The initial benefits of taking on this extra burden is the true identification maybe found, and secondly, additional integrity is applied to the message.

Also in Fig. 13 a comparison can be seen between WS-Security and CTB (exchange) which shows that WS-Security had taken

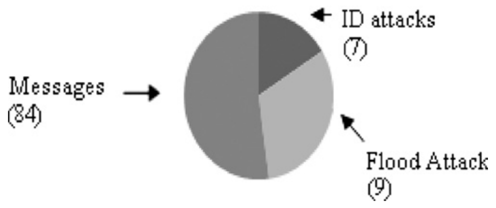


Fig. 10. 84 normal messages were processed. Nine floods were successful in crashing the system and seven attack messages were filtered during the Overload Payload attack.

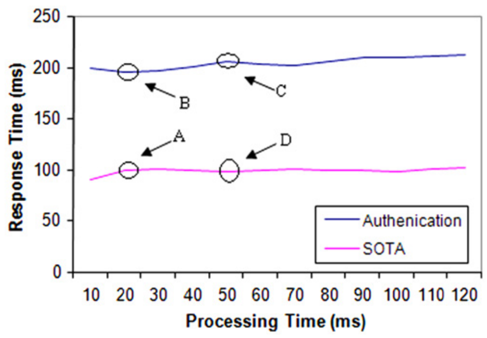


Fig. 12. Comparing the results of SOTA against the SOAP authentication.

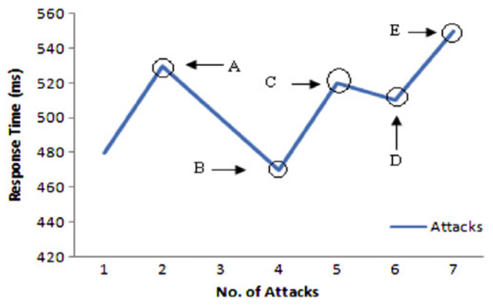


Fig. 11. The response time of CTB, in identifying the source of the attack. Point A and B shows that response time was quickened, while B to E shows an increase in response time due to 3 (points A, C and E) crashes that had taken place during identifying the attacks.

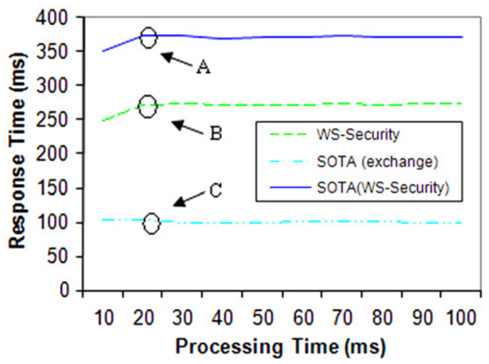


Fig. 13. Results of WS-Security, SOTA (exchange) and SOTA(WS-Security).

over twice the response time. The reasons for the increase may be many but more than likely it was due to a WS-Security built-in security token, which was placed in the message before it was sent to the Amazon Web Server. Upon receipt of the token, Amazon tested the authentication of the message, while in the comparison CTB, it only had to exchange the identification information. Assuming that Amazon had CTB on their system then traceback to the source of the attack could have occurred instead of just authenticating the message and thereby, they would have been able to identify the source of the attack.

4.6. Cloud Protector experiments

To train up the Cloud Protector there is no known dataset available on the Internet that contains normal, H-DoS or X-DoS messages. We therefore developed our own through the StuPot project (2009), and which we have already covered in previously sections of the paper.

The datasets were split into two groups, one for a training set (1000 data points) and the second as a test set (1000 data points). The Cloud Protector will firstly be trained with the trained dataset and then tested against the test dataset.

4.6.1. Evaluation of Cloud Protector for H-DoS attacks

The result of the Cloud Protector shown in Fig. 14 demonstrates that on its training sets it detected around 91% of with a miss rating of 9%. The main issue from our results was that the response time varied significantly from being able to detect the attack traffic within a matter of 20–30 ms to 1 s. One hypothesis is that the dataset was scattered far apart, and so the error ratio within the neural network kept fluctuating. Another hypothesis is that it could be the back propagation. In the next round of experiments we may attempt a HopField or Perceptron Neural Network for comparison. Another reason for the problem in variation of response times could be related to the running of the back-propagation neutral network on the VMware (for example, the settings with the memory and CPU might need slight adjusting) or some fault with the O/S image that caused type of processor corruption. All these reasons still need to be investigated further so they can be eliminated.

The scatter diagram in Fig. 15 showed the results of the trained Cloud Protector against the test dataset, with the results down by 3% (detection of 88% of attack traffic). Again the response time varied greatly, just like it did with the trained dataset. Looking at the total of the test results of the Cloud Protector, a score of 88% of the attack traffic is still good, but this score also gives us a clue that the neural network may be varying due to its setting. These are currently at 4 Neuron Layers (3, 3, 3, 1), Learning Rate of 0.2,

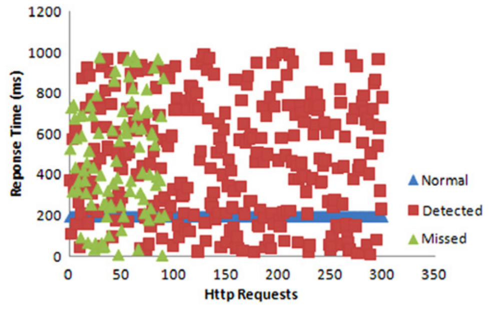


Fig. 14. The scatter diagram shows the training dataset that the Cloud Protector was trained on. As seen in experiment it was able to determine the HTTP normal traffic (100%), the detected attack traffic (91%) and the missed traffic (9%). The response time for detection varied from very good (around 20–30 ms) to very poor (1000 ms).

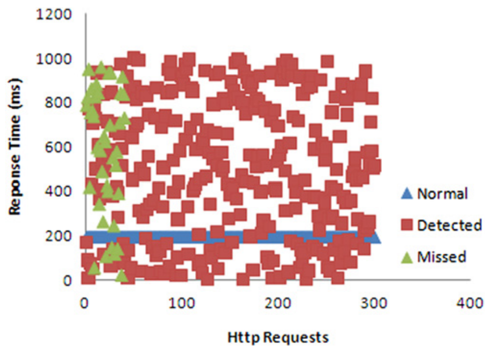


Fig. 15. The scatter diagram of the trained Cloud Protector on the test dataset which shows that it was able to detect all the normal traffic (100%), while detection was of the attack traffic was down by 3% from its training (88%). The response time for detection varied from very good (around 20–30 ms) to very poor (1000 ms).

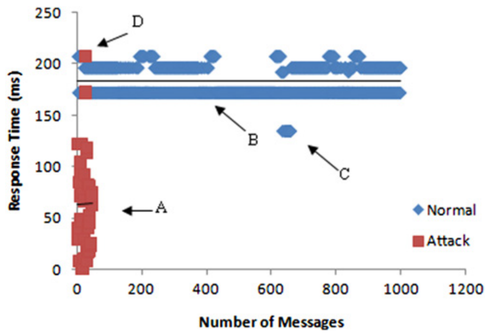


Fig. 16. The Cloud Protector results from being trained on X-DoS dataset from the StuPot Project. As it can be seen the training in the most part went quite smoothly, with a few exceptions at points C and D.

Momentum of 0.6, and a variable threshold of 0.1. Further adjustment with them may provide us with a better result.

4.6.2. Evaluation of Cloud Protector for X-DoS attacks

The results in Fig. 16 show that the training of the Cloud Protector went quite well, with the exception of points C and D. At point A, the results show that the Cloud Protector was able to detect all the attack messages, though at point D there are a few data points at around the 150–200 ms. This means that it took a little over 50 ms extra time than the other messages.

In addition to this, there is a similar pattern with normal messages in Fig. 16, as with attack messages. As can be seen at point B, detect traffic was quite uniform apart from point C, which at this particular point the normal message being detected is quite early compared to the rest of the messages. One of the reasons for this could be due to the processing speed being freed up, giving the Cloud Protector the quick burst before the O/S decided to re-assign the processor to another job.

The trained Cloud Protector is then put against the test data to see what results it can achieve. As seen in Fig. 17, it did remarkably well and in fact improved upon the training result in Fig. 16. At point A, it can be seen that the Cloud Protector was able to improve the response time of detecting attack traffic, and compared to Fig. 16, it was able to move those two data points (point D) down with the rest of the group. This also applies to normal messages. Compared to Fig. 16 at point C, it was moved up to the rest of the normal messages, though a better result would have been achieved if the messages had been reduced to the level of point C. The best result the Cloud Protector achieved at

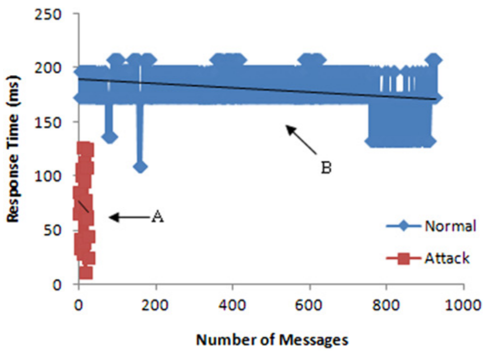


Fig. 17. The trained Cloud Protector results from being tested with the X-DoS test dataset from the StuPot Project. The results display that the trained Cloud Protector compared to Fig. 16, was able detect majority of the X-DoS data.

detecting and removing the X-DoS messages, was around 10 ms mark, while the worst result was around 135–140 ms mark.

The results from Figs. 14 and 15 compared to results from Figs. 16 and 17, show that the Cloud Protector had problems of varying response times, while the latter did have various response times but not as varied. This seems to indicate that the Cloud Protectors setting needs to be re-adjusted to handle the HTTP DoS attack traffic.

5. Conclusions and future work

According to Napper and Bientinesi (2009), the cloud computing model has the ability to scale computer resources on demand, and give users a number of advantages to progress their conventional cluster system. In addition, there is no upfront investment to update infrastructure, labour and no ongoing expenses. In fact the total cost of going towards cloud is almost zero when resources are not in use.

Therefore it is no wonder that academic research and industry are moving towards cloud computing. However, as security experts, the problem we see is recurrence of the same mistakes that were made with the development of the internet. These mistakes were related to functionality and performance which took precedence over security. Security should in fact be implemented it along side functionality and performance.

In this paper we have discussed our work on service-oriented architecture and the security application to cloud computing. We have also covered two threats that pose a very serious danger to cloud systems, H-DoS and X-DoS attacks. If one of these attacks hits the cloud, it could potentially crippling to a business like Amazon EC2.

HTTP-DoS (H-DoS) attack is potentially lethal to cloud computing because it relies on HTTP to communicate with itself and other cloud systems. As shown in this paper, we demonstrated how such an attack can take place using the same scenario that brought down the pro-Iranian websites and how it can be done within a cloud (for example, by a vindictive employee) or outside of the cloud system (for example, Bragging Rights).

We also covered another attack called Xml-Based Denial of Service (X-DoS), which is another lethal attack aimed at the services the cloud provides. To defend against such attacks, we brought forward our SOTA model and implemented it on a cloud system, which is called a Cloud TraceBack (CTB). CTB demonstrated that it can be used in an actual X-DoS attack so the cloud victim could trace the attack back to the source. Our results showed that CTB is able to find the source of an attack within a matter of seconds.

We also developed the Cloud Protector, which is a neural network that was trained to detect and filter X-DoS attacks. The result we achieved was around 98–99% of the attack traffic within an average of 10–135 ms. We also trained up the Cloud Protector on the H-DoS attack. We detected and filtered between 88% and 91%, but we achieved a very large response variance due to a problem with the neural network settings. So to solve this problem, the Cloud Protector should have two settings which interchange between HTTP message that comes in and X-DoS messages.

We plan to show through our previous research on Chaos Theory (Chonka et al., 2010), that cloud computers work on the same infrastructure as networks, thereby introducing inherent non-linear dynamics into the system so when an attacker initiates a H-DoS attack they change the initials condition of the system. We have already shown this in Chonka et al. (2010) and furthered our research into Xiang et al., 2009. We intend to further our research yet again to test whether Chaos Theory can explain H-DoS attacks on cloud computing systems.

References

Amazon Web Services. Amazon Elastic Compute Cloud (ec2), Available on the WWW, 2009. 〈http://aws.amazon.com/ec2〉, last accessed 10, November 2009.
Balding G. What everyone ought to know about cloud security. Cloudsecurity.com, 〈http://www.slideshare.net/craigbalding/what-everyone-ought-to-know-about-cloud-security〉, 2009.
Belenky A, Ansari N. Tracing multiple attackers with deterministic packet marking (DPM). In: Proceedings of IEEE Pacific Rim conference on communications, computers and signal processing, vol. 1, 2003. p. 49–52.
Chonka A, Zhou W, Xiang Y. Protecting web services with service oriented traceback architecture. In: Proceedings of the IEEE eighth international conference on computer and information technology, IEEE, 2008a.
Chonka A, Zhou W, Xiang Y. Protecting web services from DDoS attacks by SOTA. In: Proceedings of the IEEE fifth international conference on information technology and applications, IEEE, 2008b.
Chonka A, Zhou W, Xiang Y. Defending grid web services from X-DoS Attacks by SOTA. In: Proceedings of the third IEEE international workshop on web and pervasive security (WPS 2009), IEEE, 2009.
Chonka A, Zhou W, Singh J. Chaos theory based detection against network mimicking DDoS attacks. IEEE Communications Letters 2010;13(9):717–9.
Danchev D. Iranian opposition launches organized cyber attack against pro-Ahmadinejad sites. ZDNet blog, 〈http://blogs.zdnet.com/security/?p=3613〉, June 15, 2009a.
Danchev D. Iranian Opposition DDoS-es pro Ahmadinejad Sites. Dancho Danchev's Blog, 〈http://ddanchev.blogspot.com/2009/06/iranian-opposition-ddos-es-pro.html〉, 16 June 2009b.
Dean D. An algebraic approach to IP traceback. ACM Transactions on Information and System Security (1094–9224) 2002;5(2):119.
Dittrich D. The "mstream" distributed denial of service attack tool. University of Washington, 〈http://staff.washington.edu/dittrich/misc/mstream.analysis.txt〉, 2000.
Dittrich D. The DoS Project's "trinoo" distributed denial of service attack tool. University of Washington, 〈http://staff.washington.edu/dittrich/misc/trinoo.analysis〉, 1999.
E-Crime Congress. E-Crime Survey, 〈http://www.e-crimecongress.org/ecrime2009/documents/e-CrimeSurvey2009_AKJ_KPMG(1).pdf〉 2009.
ElasticHosts. Cloud hosting, Available on WWW, 〈http://www.elastichosts.com/?gclid=CMSSxpyAjJ4CFcEtpAodOyJRqw〉, 2009.
Enomaly.com. Enomalism elastic computing infrastructure. 〈http://www.enomaly.com〉, 2009.
F-Secure. F-Secure Virus Descriptions: Agobot. 〈http://www.f-secure.com/v-descs/agobot.shtml〉, 2003.
Fowler A. Fear in Fast Lane. Four Corners Program, ABC, 〈http://www.abc.net.au/4corners/content/2009/s2655088.htm〉, 2009.
Georgia Government. Russian Cyberwar on Georgia. 〈www.georgiaupdate.gov.ge〉, October 2008, lasted accessed 17 July 2009.
He Y, Chen W, Peng, W, Yang, M. Efficient and beneficial defence against DDoS direct attack and reflector attack, ISPA, 2005, LNCS, vol. 3758, p. 576–87.
Hewlett-Packard. HP integrated lights-out 2 user guide. Technical report, HP, 2009.
Hibler M, Ricci R, Stoller, L Duerig J, Guruprasad S, Stack T, et al. Large-scale virtualization in the emulab network testbed. In: Proceedings of the 2008 USENIX annual technical conference, 2008.
INetu Managed Hosting. Cloud Hosting, Available on WWW, 〈http://www.inetu.net/hosting-solutions/flexible-capacity.php?INCC=Adwords.hybrid&gclid=CLC844r_i54CFc0vpAoduizPqA〉, 2009.
Jensen M, Gruschka N, Herkenhöner R, Luttenberger N. SOA and web services: new technologies, new standards – new attacks. In: Proceedings of the fifth European conference on web services, 0-7695-3044-3/07, 2007.

Keahey K, Foster I, Freeman T, Zhang X. Virtual workspaces: achieving quality of service and quality of life in the grid. Science Program 2005;13(4):265–75.

Laplante P, Zhang J, Voas J. What's in a name? Distinguishing between saas and soa IT Professional 2008;10(3):46–50.

Lenk A, Klems M, Nimis J, Tai S, Sandholm T. What's inside the Cloud? An architectural map of the Cloud landscape. In: International Conference on Software Engineering, Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009. p. 23–31, ISBN:978-1-4244-3713-9.

McNett M, Gupta D, Vahdat A, Voelker GM, Usher: an extensible framework for managing clusters of virtual machines. In: Proceedings of the 21st Large installation system administration conference (LISA), November 2007.

Metz C. DDoS attack rains down on Amazon cloud. The Register, Online Article, 〈http://www.theregister.co.uk/2009/10/05/amazon_bitbucket_outage/〉, 26 October 2009.

Nadalin A, Kaler C, Monzillo R, Hallam-Baker P. Web Services Security: SOAP Message Security 1.1 (WSSecurity 2004). 〈http://docs.oasis-open.org/wss/v1.1/〉, 2008a.

Nadalin A, Kaler C, Monzillo R, Hallam-Baker P. Web Services Security: SOAP Message Security 1.1 (WSSecurity 2004). 〈http://docs.oasis-open.org/wss/v1.1/〉, 2008b.

Napper J, Bientinesi P. Can Cloud Computing Reach the Top500? UCHPC-MAW'09, Ischia, Italy, May 18–20, 2009.

Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, et al. The eucalyptus open-source cloud-computing system. In: EEE Computer Society, CCGRID'09: Proceedings of the 2009 ninth IEEE/ACM international symposium on cluster computing and the grid, 2009. p. 124–31.

Padmanabhuni S; Singh V. Senthil kumar KM, Chatterjee A. Web services, preventing service oriented denial of service (PreSODoS): a proposed approach. In: Proceedings of the ICWS apos;06, international conference on volume , issue, September 2006. p.577–84.

PageRebooter.com, 〈http://www.pagereboot.com/〉, last accessed 10 November 2009.

Rogers L. What is a Distributed Denial of Service (DDoS) attack and what can I do about it? Computer Emergency Response Team, 〈http://www.cert.org/homeusers/ddos.html〉, last accessed 19 November 2009.

Savage S, Wetherall D, Karlin A, Anderson T. Practical network support for IP traceback. SIGCOMM'00, Stockholm, Sweden 2000, 2001.

Sophos, 〈http://www.sophos.com〉, 2009.

Stewart J. HTTP DDoS attack mitigation using tarpitting. Securework.com, 〈http://www.secureworks.com/research/threats/ddos/〉, June 25, 2007, last accessed 20, 2010.

TCPDump.com, 〈http://www.tcpdump.org/〉 last accessed 10 November, 2009.

The Study of Honeypot Technology (StuPot). Deakin University, 〈http://www.deakin.edu.au/~ashley/〉, 2009.

Wireshark.com. t-shark, 〈http://www.wireshark.org/docs/man-pages/tshark.html〉, accessed on 10 November 2009.

van Beignum I. *I*Pv6: coming to a root server near you. ars technical, 〈http://arstechnica.com/news.ars/post/20080102-icann-to-add-ipv6-addresses-for-root-dns-servers.html〉, 02 January 2008.

Vmware.com. VMware Server 2, 〈http://www.vmware.com/products/server/〉, 10 November 2009.

Xiang X, Zhou W, Guo M. Flexible deterministic packet marking: an IP traceback system to find the real source of attacks. IEEE Transactions on Parallel and Distributed Systems 2009;20(4):567–80.

Ye X, Singh S. A soa approach to DDoS attacks. In: Proceedings of the IEEE international conference on web services (ICWS 2007), 2007. pp. 567–74.