International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume III, Issue V, May- 2016

Monitoring and Prevention of Stealthy Denial of Service Strategy in Cloud Computing

Ms.S.Priyanka
Department of Computer Science and Engineering
SSM College of Engineering, Komarapalayam,
Tamil Nadu, India
Priyankacse1701@gmail.com

Ms.S.Vimalananthi
Department of Computer Science and Engineering
SSM College of Engineering, Komarapalayam
Tamil Nadu, India
vimalananthis@yahoo.com

Abstract—The success of the cloud computing paradigm is due to its on-demand, self-service, and pay-by-use nature. According to this paradigm, the effects of Denial of Service (DoS) attacks involve not only the quality of the delivered service, but also the service maintenance costs in terms of resource consumption. Specifically, the longer the detection delay is, the higher the costs to be incurred. Therefore, a particular attention has to be paid for stealthy DoS attacks. They aim at minimizing their visibility, and at the same time, they can be as harmful as the brute-force attacks. They are sophisticated attacks tailored to leverage the worst-case performance of the target system through specific periodic, pulsing, and low-rate traffic patterns. In this paper, we propose a strategy to orchestrate stealthy attack patterns, which exhibit a slowly-increasing-intensity trend designed to inflict the maximum financial cost to the cloud customer, while respecting the job size and the service arrival rate imposed by the detection mechanisms. We describe both how to apply the proposed strategy, and its effects on the target system deployed in the cloud.

Index Terms—Cloud computing, sophisticated attacks strategy,low-rate attacks, intrusion detection

I-INTRODUCTION

Cloud Computing is an emerging paradigm that allows customers obtain cloud resources and services according to an on-demand, self-service, and pay-by-use business model. Service level agreements (SLA) regulate the costs that the cloud customers have to pay for the provided quality of service (QoS). A side effect of such a model is that, it is prone to Denial of Service (DoS) and Distributed DoS (DDoS), which aim at reducing the service availability and performance by exhausting the resources of the serv-ice's host system (including memory, processing resources, and network bandwidth). Such attacks have special effects in the cloud due to the adopted pay-by-use business model. Specifically, in cloud computing also a partial ser-vice degradation due to an attack has direct effect on the service costs, and not only on the performance and avail-ability perceived by the customer. The delay of the cloud service provider to diagnose the causes of the service degra-dation (i.e., if it is due to either an attack or an overload) can be considered as a security vulnerability. It can be exploited by attackers that aim at exhausting the cloud resources (allocated to satisfy the negotiated QoS), and seriously degrading the QoS, as happened to the BitBucket Cloud, which went down for 19h. Therefore, the cloud manage-ment system has to implement specific countermeasures in order to avoid paying credits in case of accidental or deliberate intrusion that cause violations of QoS guarantees. Over the past decade, many efforts have been devoted to the detection of DDoS attacks in distributed systems. Secu-rity prevention mechanisms usually use approaches based On rate-controlling, time-window, worst-case threshold, and pattern-matching methods to discriminate between the nominal system operation and malicious behaviors . On the other hand, the attackers are aware of the presence of such protection mechanisms.

They attempt to perform their activities in a "stealthy" fashion in order to elude the secu-rity mechanisms, by orchestrating and timing attack pat-terns that leverage specific weaknesses of target systems . They are carried out by directing flows of legitimate service requests against a specific system at such a low-rate that would evade the DDoS detection mechanisms, and prolong the attack latency, i.e., the amount of time that the ongoing attack to the system has been undetected.

This paper presents a sophisticated strategy to orches-trate stealthy attack patterns against applications running in the cloud.

Instead of aiming at making the service unavail-able, the proposed strategy aims at exploiting the cloud flex-ibility, forcing the application to consume more resources than needed, affecting the cloud customer more on financial aspects than on the service availability. The attack pattern is orchestrated in order to evade, or however, greatly delay the techniques proposed in the literature to detect low-rate attacks. It does not exhibit a periodic waveform typical of low-rate exhausting attacks. In contrast with them, it is an iterative and incremental process. In particu-lar, the attack potency (in terms of service requests rate and concurrent attack sources) is slowly enhanced by a patient attacker, in order to inflict significant financial losses, even if the attack pattern is performed in accordance to the maxi-mum job size and arrival rate of the service requests allowed in the system. Using a simplified model empirically designed, we derive an expression for gradually increasing the potency of the attack, as a function of the reached service degradation (without knowing in advance the target system capability). We show that the features offered by the cloud provider, to ensure the SLA negotiated with the customer (including the load balancing and auto-scaling mecha-nisms), can be maliciously exploited by the proposed stealthy attack, which slowly exhausts the resources pro-vided by the cloud provider, and increases the costs incurred by the customer.

The proposed attack strategy, namely Slowly-Increasing-Polymorphic DDoS Attack Strategy (SIPDAS) can be applied to several kind of attacks, that leverage known application vulnerabilities, in order to degrade the service provided by the target application server running in the cloud. The term polymorphic is inspired to polymorphic attacks which change message sequence at every successive infection in order to evade signature detection mechanisms. Even if the victim detects the SIPDAS attack, the attack strategy can be re-initiate by using a different application vulnerability (polymorphism in the form), or a different timing (polymor-phism over time).

In order to validate the stealthy characteristics of the proposed SIPDAS attack, we explore potential solutions proposed in the literature to detect sophisticated low-rate DDoS attacks. We show that the proposed slowly-increas-ing polymorphic behavior induces enough overload on the target system (to cause a significant financial losses), and evades, or however, delays greatly the detection methods. Moreover, in order to explore the attack impact against an application deployed in a cloud environment, this paper focuses on one of the most serious threats to cloud computing, which comes from XML-based DoS (X-DoS) attacks to the web-based systems . The experi-mental testbed is based on the mOSAIC framework, which offers both a 'Software Platform', that enables the execution of applications developed using the mOSAIC API, and a 'Cloud Agency', that acts as a provisioning system, brokering resources from a federation of cloud providers.

The rest of this paper is organized as follows. Back-ground and related work are presented. Illustrates several examples of attacks, which can be leveraged to implement the proposed attack pattern. Describes the proposed strategy to build the stealthy attacks, and presents the attack pattern, whose detailed implementation is reported. Introduces the X-DoS attack used as case study. It shows the experimental results obtained running the attack pattern. Validation of stealthy characteristics is provided. Some considerations about coun-termeasures against the proposed strategy are illustrated. Conclusions and future work are described .

II-RELATED WORKS

INTRUSION DETECTION IN THE CLOUD

Intrusion Detection Systems (IDS) have been used widely to detect malicious behaviors in network communication and hosts. IDS management is an important capability for distributed IDS solutions, which makes it possible to integrate and handle different types of sensors or collect and synthesize alerts generated from multiple hosts located in the distributed environment. Facing new application scenarios in Cloud Computing, the IDS approaches yield several problems since the operator of the IDS should be the user, not the administrator of the Cloud infrastructure. Extensibility, efficient management, and compatibility to virtualization based context need to be introduced into many existing IDS implementations. Additionally, the Cloud providers need to enable possibilities to deploy and configure IDS for the user.

They summarize several requirements for deploying IDS in the Cloud and propose an extensible IDS architecture for being easily used in a distributed cloud infrastructure.

Along with the proposal of the concept Cloud Computing, a new paradigm of software development and deployment has emerged. Cloud Computing can be interpreted as the sum of Software as a Service (SaaS) and Utility Computing. There are Cloud providers, which offer a specific virtualized infrastructure, and Cloud users, which use the provided services and infrastructure. Furthermore, there are three layers involved in Cloud Computing: the system layer, the platform layer, and the application layer. The hardware layer is

the basis for Cloud computing and is not provided directly to the user.

CLOUD SECURITY DEFENCE TO PROTECTING CLOUD COMPUTING AGAINST HTTP-DOS AND XML-DOS ATTACKS

Today it is an inescapably fact that the internet and cloud computing are the future revenue generators for businesses and corporations, hence why organization like IBM are heavily investing in providing better service oriented products to facilitate the demands. One of the most serious threats to these future revenues and to cloud computing itself comes from HTTP Denial-of-Service or XML Based Denial of Service attack. These types of attacks are simple and easy to implement by the attacker but to security experts they are twice as difficult to stop. They recreate some of the current attacks that attackers may initiate as HTTP and XML and offer a possible solution to trace back through their Cloud Trace Back (CTB) to find the source of these attacks as well as to introduce the use of a back propagation neutral network, called Cloud Protector, that was trained to detect and filter such attack traffic. Their results show that they were able to detect and filter around an average of 100% of the attack messages and were able to identify the source of the attack within 10 seconds.

Today, cloud computing systems are providing a wide variety of services and interfaces to which vendors are now renting out spaces on their physical machines at an hourly rate for a tidy profit (Amazon EC2, 2009; INetu, 2009 & Elastic Hosts, 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 and McNettet. al. 2007) to flexible hosted software services (Laplante et. al. 2008; Hewlett-Packard, 2009; Hibler et. al. 2008 and Lenk et. al, 2009) and each shares the notion that delivered resources are to be allocated and de-allocated on demand, while at the same time provide reasonable performance.

DETECTING APPLICATION DENIAL-OF-SERVICE ATTACKS: A GROUP TESTING BASED APPROACH

Application DoS attack, which aims at disrupting application service rather than depleting the network resource, has emerged as a larger threat to network services, compared to the classic DoS attack. Owing to its high similarity to legitimate traffic and much lower launching overhead than classic DDoS attack, this new assault type cannot be efficiently detected or prevented by existing detection solutions. To identify application DoS attack. They propose a novel group testing (GT) based approach deployed on back-end servers, which not only offers a theoretical method to obtain short detection delay and low false positive/negative rate, but also provides an underlying framework against general network attacks.

More specifically, they first extend classic GT model with size constraints for practice purposes, then re-distribute the client service requests to multiple virtual servers embedded within each backend server machine, according to specific testing matrices. Base on this framework, they propose a 2-mode detection mechanism using some dynamic thresholds to efficiently identify the attackers. The focus of this work lies in the detection algorithms proposed and the corresponding theoretical complexity analysis. They also provide preliminary simulation results regarding the efficiency and practicability of this new scheme. Further discussions over implementation issues and performance enhancements are also appended to show its great potentials.

Denial-of-Service (DoS) attack, which aims to make a service unavailable to legitimate clients, has become a severe threat to the Internet security . Traditional DoS attacks mainly abuse the network bandwidth around the Internet subsystems and degrade the quality of service by generating congestions at the network . Consequently, several network based defense methods have tried to detect these attacks by controlling traffic volume or differentiating traffic patterns at the intermediate routers. However, with the boost in network bandwidth and application service types recently, the target of DoS attacks have shifted from network to server resources and application procedures themselves, forming a new application DoS attack.

INTRUSION TOLERANCE OF STEALTH DOS ATTACKS TO WEB SERVICES

Focuses on one of the most harmful categories of Denial of Service attacks, commonly known in the literature as “stealth” attacks. They are performed avoiding to send significant volumes of data, by injecting into the network a low-rate flow of packets in order to evade rate-controlling detection mechanisms. This work presents an intrusion tolerance solution, which aims at providing minimal level of services, even when the system has been partially compromised by such attacks. It describes all protection phases, from monitoring to diagnosis and recovery. Preliminary experimental results show that the proposed approach results in a better performance of Intrusion Prevention Systems, in terms of reducing service unavailability during stealth attacks.

Denial of Service (DoS) attacks are serious threats to the Internet causing billions of dollars in economic loss. In particular, brute force and flooding attacks against application-layer services, like the Web Services (WSs), pose a huge risk to several business-critical services.

The recent tide of DoS attacks against high-profile WSs, including PayPal, MasterCard and Amazon, demonstrate how devastating DoS attacks are. In general, attackers are aware of the presence of protection mechanisms: they thus attempt to perform their activities in a stealthy fashion in order to elude local security mechanisms. From an attacker point of view, one of the most effective way of circumventing these security countermeasures consists in distributing the attacks both in 'form' and/or 'time' executing.

INTRUSION TOLERANT APPROACH FOR DENIAL OF SERVICE ATTACKS TO WEB SERVICES

Intrusion Detection Systems are the major technology used for protecting information systems. However, they do not directly detect intrusion, but they only monitor the attack symptoms. Therefore, no assumption can be made on the outcome of the attack, no assurance can be assumed once the system is compromised. The intrusion tolerance techniques focus on providing minimal level of services, even when the system has been partially compromised. This paper presents an intrusion tolerant approach for Denial of Service attacks to Web Services.

It focuses on the detection of attack symptoms as well as the diagnosis of intrusion effects in order to perform a proper reaction only if the attack succeeds. In particular, this work focuses on a specific Denial of Service attack, called Deeply-Nested XML. Preliminary experimental results show that the proposed approach results in a better performance of the Intrusion Detection Systems, in terms of increasing diagnosis capacity as well as reducing the service unavailability during an intrusion.

Intrusion tolerance is an emerging paradigm for developing systems that are able to provide an acceptable level of service even after that intruders have broken in. In the context of an Intrusion Tolerant System (ITS), intrusion detection is the key activity to perform intrusion recovery. It is characterized by two sub-activities: monitoring and diagnosis. Monitoring recognizes that something unexpected has occurred in the system.

III-EXISTING SYSTEM METHODOLOGY

In order to implement DoS-based attacks, the following components are involved:

- A Master that coordinates the attack;
- πAgents that perform the attack (each Agent injects a single flow of messages ϕAj); and
- A Meter that evaluates the attack effects.

The existing system consists of the approach where each Agent performs a stealthy service degradation in the cloud computing. It has been specialized for an X-DoS attack. Specifically, the attack is performed by injecting polymorphic bursts of length T with an increasing intensity until the attack is either successful or detected. Each burst is formatted in such a way as to inflict a certain average level of load C_R.

That is a web service is called continuously or a file/image is accessed continuously by the same client.

In particular, it is assumed that C_R is proportional to the attack intensity of the flow ϕAj during the period T. Therefore, denote I_0 as the initial intensity of the attack, and assuming $\Delta CR = \Delta I$ as the increment of the attack intensity.

For each attack period, fixed the maximum number of nested tags (tagThreshold), the routine pickRandomTags(...) randomly returns the number of nested tags nT for each message (row 4). Based on n_T, the routine computeInterarrivalTime uses a specific algorithm Eq. (1) to compute the inter-arrival time for injecting the next message.

$$C_R(n_T, t_I) = \alpha t_I^{-\beta} * e^{(\varepsilon n_T)} \Rightarrow t_I = \sqrt[\beta]{\frac{\alpha e^{(\varepsilon n_T)}}{C_R}},$$

At the end of the period T, if the condition 'attackSuccessful' is false, the attack intensity is increased . If the condition 'attackSuccessful'is true, the attack intensity is maintained constant until either the attack is detected (e.g., the target system is no longer reachable due to a reaction performed by the security administrator or by an Intrusion Prevention System), or the auto-scaling mechanism enabled in the cloud adds new cloud resources. The attack is performed until it is either detected, or the average message rate of the next burst to be injected is greater than δ_T.

- Only attack scenarios are considered.
- Not aimed at extending the approach to a larger set of application level vulnerabilities.
- Prevention of those attack mechanisms is not studied.
- Security level of existing system is very low, maintained data may get lost or theft by the unauthorized users.
- The frequently requested web pages, images, and most requested clients are not trace out quickly.

IV-PROPOSED SYSTEM METHODOLOGY

In addition to the existing system implementation, the proposed system also provides an environment where attack scenario is found out and prevented.The resources such as web pages/ images and web services are added in a database with access count and time limit. For example, a particular resource can be accessed hundred times within a hour by one particular client IP address. If the client accesses the resource more than the given count, the request is redirected to a 'accessdenied' page.

Both attack scenarios and prevention approach are considered.

- Aimed at extending the approach to a larger set of application level vulnerabilities.
- Prevention of those attack mechanisms is studied.
- Security level of existing system is very high, maintained data is not lost or theft by the unauthorized users.
- The frequently requested web pages, images, and most requested clients are traced out quickly.
- At the same time, privileged clients can access the resources from their web site coding also.

V-IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

MODULES:

1. Detecting The Polymorphic Attacks In The Network
2. Ipaddress Blocking
3. Resources Settings to Be Monitored For DDOS Attack
4. Monitor And Prevent The Ddos
5. Request Log And Captcha Form
6. Check For Web Referal Architecture Based Navigation

1. DETECTING THE POLYMORPHIC ATTACKS IN THE NETWORK

In this module, the proposed attack strategy, namely Slowly-Increasing-Polymorphic DDoS Attack Strategy (SIPDAS) is applied. It leverages known application vulnerabilities, in order to degrade the service provided by the target application server running in the cloud.

The term polymorphic is inspired to polymorphic attacks which change message sequence at every successive infection in order to evade signature detection mechanisms. The attack is performed until it is either detected.

ALGORITHM : CORE ALGORITHM OF SIPDAS AGENT

```
Require: Integer timeWindow (T {Burst period.}
Require: Integer nT (0 {Nested tags within each
message.}
Require: Integer tagThresold (NT {Nested tags
threshold.}
Require: Integer rateThreshold (DT {Attack rate
threshold.}
Require: Integer attackIncrement (DI {Attack intensity
increment.}
Require: Integer CR (I0 {Initial attack intensity.}
 repeat
t ( 0;
while t T do
nT ( pickRandomTagsðtagThresoldÞ;
tI ( computeInterarrivalTimeðCR; nTÞ;
sendMessageðnT ; tIÞ;
 t ( t þ tI;
 end while
 if !ðattackSuccessfulÞ then
CR ( iCR ) attackIncrement); {Attack intensification}
 else
while !ðattack detectedÞ and attackSuccessful do
{Service degradation achieved; attack intensity is fixed}
 nT ( pickRandomTagsðtagThresoldÞ;
 tI ( computeInterarrivalTimeðCR; nTÞ;
 sendMessageðnT ; tIÞ;
 end while
 end if
 tI MðCRÞ ¼ computeInterarrivalTimeðCR; NTÞ;
 tI mðCRÞ ¼ computeInterarrivalTimeðCR; 1Þ;
 until ð2=□ tIM  tImÞ < rateThresholdÞ and !ðattack detectedÞ
 if attack detected then
{Notify to the Master that the attack has been detected}
 print 0Attack detected0;
 else
{Notify to the Master the attack has reached the threshold dT
and archived the intensity CR ¼ CRM }
print 0Threshold reached0;
{Continue the attack by using the previous CR value}
```

CR ¼ CR attackIncrement;
loop
nT (pickRandomTagsðtagThresoldÞ;
tI (computeInterarrivalTimeðCR; nTÞ;
sendMessageðnT ; tIÞ;
end loop
end if

2. IPADDRESS BLOCKING

In this module, the clients IPAddress details to be blocked are added in back end table. Any IPAddress can be added or removed at any time. During addition, listening this address for all page requests or particular page request is selected. If particular page, then page URL is given. The minimum number of request count and time is entered so that only after that limit is reached, the request is redirected.

IPClassifier classifies all inbound packets into three categories: packets addressing the website's privilege port which are dropped, TCP packets which are forwarded to IPVerifier, and other packets, such as UDP and ICMP, which are forwarded to the normal forwarding path.

3. RESOURCES SETTINGS TO BE MONITORED FOR DDOS ATTACK

In this module, the source web pages such as html or aspx page are entered. In addition, image files such as jpg or gif files path is entered so that they can be listened for attacks.Therefore, supposing that mRð#kÞ and sRð#kÞ are the mean and standard deviation of the response time tR for the messages type #k, empirically estimated during the training phase, the Meter can adopt the following Chebyshev's inequality to compute deviation of the service time tSð'iÞ during the attack:

$$p(|t_S(\varphi_i) - \mu_R(\vartheta_k)| \geq \lambda * \sigma_R(\vartheta_k)) \leq \frac{1}{\lambda^2} \quad with \quad \varphi_i \in \vartheta_k.$$

4. MONITOR AND PREVENT THE DDOS

In this module, the global.asax (Active Server Application) page is written with attack listening coding. The requested client URL's IPAddress is checked whether it is blocked. If that particular client is requesting more than given specified times with in given time period.

In this module, attack prevention coding is written such that requested client URL's IPAddress is checked whether it is blocked. If that particular client is requesting more than given specified times with in given time period then it is redirected to accessdenied.aspx page.

5. REQUEST LOG AND CAPTCHA FORM

In this module, the requests made by clients are saved for future analysis. The records are displayed using GridView control which is bind through DataAdapter. In this module, a web page is designed with CAPTCHA form, in which, the mathematical equation is randomly generated and after solving the equation, the required web page is navigated.

IPVerifier verifies every TCP packet's capability token embedded in the last octet of the destination IP address and the 2-octet destination port number. Verification of a packet invokes the MAC over a 5-byte input and a 64-bit secret key. The packets carrying correct capability tokens are sent to IPRewrite, which sets a packet's destination IP to that of the target website and destination port to port. WRAPS overcome the drawbacks through checking the HTTP_REFERER property in Request. If the value is null, it is clear that the page is requested programmatically by an application.

6. CHECK FOR WEB REFERAL ARCHITECTURE BASED NAVIGATION

In this module, a web request is checked such that the router application receives the request, process the query string information, the ip address parsing work done and the request is authenticated.

VI-CONCLUSION

In this project, it proposed an effective and efficient IP traceback scheme against DDoS attacks based on entropy variations. It is a fundamentally different traceback mechanism from the currently adopted packet marking strategies. Many of the available work on IP traceback depend on packet marking, either probabilistic packet marking or deterministic packet marking. Because of the vulnerability of the Internet, the packet marking mechanism suffers a number of serious drawbacks: lack of scalability; vulnerability to packet pollution from hackers and extraordinary challenge on storage space at victims or intermediate routers.

On the other hand, the proposed method needs no marking on packets, and therefore, avoids the inherent shortcomings of packet marking mechanisms. It employs the features that are out of the control of hackers to conduct IP traceback. It observes and store short-term information of flow entropy variations at routers. Once a DDoS attack has been identified by the victim via detection algorithms, the victim then initiates the pushback tracing procedure.

REFERENCES

[1] F. Cheng and C. Meinel, "Intrusion Detection in the Cloud," in Proc. IEEE Int. Conf. Dependable, Autonom. Secure Comput., Dec. 2009, pp. 729–734

[2] M. Armbrust, A. Fox, R. Griffith, et al.: Above the Clouds: A Berkley View of Cloud Computing, Website: http://radlab.cd.berkley.edu/, UC Berkley Reliable Adaptive Distributed Systems Laboratory (Feb. 2009)

[3] Google App Engine, Website: http://code.google.com/appengine/, Google (accessed on Oct 2009)

[4] Windows Azure Platform, Website: http://www.microsoft.com/azure/, Microsoft Corporation (accessed on Oct 2009)

[5] Amazon Elastic Compute Cloud (Amazon EC2) Website: http://aws.amazon.com/ec2/, Amazon (accessed on Oct 2009)

[6] Laureano, M., Maziero, C., Jamhour, E.: Protecting host-based intrusion detectors through virtual machines Computer Networks: The International Journal of Computer and Telecommunications Networking, vol.51, Issue 5, pp. 1275-1283 (April 2007)

[7] F-Secure Linux Security, Website: http://www.f-secure.com/linuxweblog/, F-Secure Corporation (accessed Oct 2009)

[8] Samhain IDS, Website: http://www.la-samhna.de/samhain/ (accessed Oct 2009)

[9] Snort IDS, Website: http://www.snort.org/ (accessed Oct 2009)

[10] Prelude IDS, Website: http://www.prelude-ids.com/, PreludeIDS Technologies (accessed Oct 2009)