IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 7, September 2014.
www.ijiset.com
ISSN 2348 – 7968

DNIDPS: Distributed Network Intrusion Detection and Prevention System

S.Subapriya[1], Ms.N.Radhika[2]
[1] P.G Student, Computer Science Department, Prist University, Trichirapalli, TamilNadu, India
[2] Assistant Professor, Computer Science Department, Prist University Trichirapalli, TamilNadu, India
subapriya1987@gmail.com,rakshaacse@gmail.com

Abstract

Cloud security is one of most important issues that have attracted a lot of research and development effort in past few years. The Distributed Denial of Service (DDOS) attacks caused by the vast flow of requests from various clients to the cloud server at the same time. In this paper proposed a Distributed Network Intrusion Detection and Prevention System (DNIDPS) an efficient model termed as the Intrusion Detection and Prevention System (IDPS) which combines both IDS and IPS in a single mechanism by using Hashed Based Pattern Matching Algorithm and hybrid of both Anomaly Detection (AD) and Signature Based Detection. It captures and inspects suspicious cloud traffic without interrupting user's applications, cloud services and various DDOS attacks. This is very much reduced in DNIDPS and also less CPU Utilization, less virtual machine creation time, less Infrastructure Response Time (IRT) and memory consumption.

Keywords: DDOS**,** Network Intrusion Detection and Prevention System (NIDPS), Hash Based Pattern Algorithm, Anomaly Detection (AD) and Signature Based Detection.

I .Introduction

A recent CSA (Cloud Survey Alliance) survey reports that among all security issues exploitation and despicable use of cloud computing is considered as the main security threat. In traditional data centers, where system administrators have full control over the host machines, vulnerabilities can be detected and patched by the system administrator in a centralized manner. However, patching known security holes in cloud data centers, where cloud users usually have the privilege to control software installed on their managed VMs, may not work effectively and can violate the Service Level Agreement (SLA). Furthermore, cloud users can install vulnerable software on their VMs, which essentially contributes to loopholes in cloud security. The challenge is to establish an effective vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users. The vital attack to be prevented is Distributed Denial of Service (DDOS) attacks in cloud computing environment. One of the efficient methods for detecting DDOS is to use the Intrusion Detection and Prevention Systems (IDPS), in order to assure usable cloud computing services. DDOS attacks usually involve early stage actions such as multistep exploitation, low-frequency vulnerability scanning, and compromising identified vulnerable virtual machines as zombies and the detection of zombie exploration attacks is Extremely difficult. we propose a distributed vulnerability detection, measurement and countermeasure selection mechanism which is built on attack graph-based analytical

Models and reconfigurable virtual network-based countermeasures.

Thus, this work utilizes both systems: (IDS) and (IPS) and refers to it as Intrusion Detection and Prevention System (IDPS). Furthermore, many works have been done in using one of the (IDS) techniques; either Anomaly Detection (AD) or Signature based Detection or hybrid of both. NIPS systems are usually comprised of two major components: a pattern matching engine and a packet classification engine. The pattern matching engine's input is a packet and its output is a set of matched patterns belonging to the set of well known attack's signatures. Content addressable memories (CAMs) minimize the number of memory accesses required to locate the entry. Given an input key, the CAM device compares it against all memory words in parallel; hence, a lookup procedure requires one clock cycle. Unlike standard computer memory (RAM) in which the user supplies a memory address and the RAM returns the data word stored at that address, a CAM is designed such that the user supplies a data word and the CAM searches its entire memory to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found (and in some Architectures, it also returns the data word, or other associated pieces of data). TCAM Based Implementation Of Multi Pattern Matching Using Covered State Encoding uses a state encoding scheme called a covered state encoding for the efficient TCAM-based implementation of the Aho-Corasick multi pattern matching algorithm, which is used in network intrusion detection systems character processing.

ISSN 2348 – 7968

II. RELATEDWORK

DIDS a proposed approach optimizes the implementation on cloud servers to minimize resource consumption. DIDS includes two main phases; deploy a lightweight mirroring Deep Packet Inspection (DPI) is applied and virtual network reconfigurations can be deployed to the inspecting Virtual Machine (VM) to make the potential attack behaviors prominent.

Firewall security, like any other technology, requires proper management in order to provide proper security services. Thus, just having firewalls on the network boundaries or between sub-domains may not necessarily make the network any secure. One reason of this is the complexity of managing firewall rules and the resulting network vulnerability due to rule anomalies. The Firewall Policy Advisor presented provides a number of techniques for purifying and protecting the firewall policy from rule anomalies. The administrator may use the firewall policy advisor to manage legacy firewall policies without prior analysis of filtering rules. It is formally defined a number of firewall policy anomalies in both centralized and distributed firewalls and these are the only conflicts that could exist in firewall policies have been proved. It presented a set of algorithms to detect rule anomalies within a single firewall (intra-firewall anomalies), and between inter-connected firewalls interfirewall anomalies) in the network. When an anomaly is detected, users are prompted with proper corrective actions. A tool is intentionally made not to automatically correct the discovered anomaly but rather alarm the user because it is believed that the administrator should have the final call on policy changes.

Finally, a user-friendly java-based implementation of Firewall Policy Advisor is presented. Using Firewall Policy Advisor was shown to be very effective for real-life networks. In regards to usability, the tool was able to discover filtering anomalies in rules written by expert network administrators. In regards to performance, although the policy analysis algorithms are paradoxically dependant on the number of rules in the firewall policy, our experiments show that the average processing time in intra- and inter-firewall anomaly discovery is very reasonable for practical applications. The technical contributions are three-fold. First, formally a firewall tree is specified. Second, two classes of properties, namely accept and discard properties, of firewall trees are identified. Third, two algorithms that can be used to verify whether any given firewall tree satisfies a given accept or discard property of that tree.

Firewalls are core Alert Correlation algorithm is used for constructing the attack graph which in turn is used to define the possible vulnerable paths in the network. If a particular alert is new, a specific edge in the attack graph is created which leads to new set of alerts. The output of Alert Correlation Graph is the set of attack paths. This module uses Active Alert graph verification algorithm .The Entropy calculation is used for calculating the amount of traffic. Sometimes the intruder may impose DDOS attack/flooding attack. So the system which is implementing more traffic needs to be tracked. The monitoring process can be done to calculate the behavioral distance. Abnormal behavior of the client is frequently reported to the administrator. Also the countermeasures are selected to perform which include traffic redirection, port blocking, network reconfiguration, updates the filtering rules, Deep packet inspection, and Virtual Machine isolation.

III. Distributed Network Intrusion Detection and Prevention System Components.

DIDPS employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective counter measures. DIDPS is used to identify the source or target of the intrusion to detect multistep attack. It also establishes a defense-in-depth intrusion detection and prevention framework to mitigate DDOS attacks in cloud environment. The DDOS attacks are prevented in DIDPS.

A. Decentralized Access Control

It is based on intrusion detection techniques. Here to Merge Entropy based System with Anomaly detection System for providing multilevel Distributed Denial of Service (DDOS). This entropy takes the peak and off peak traffic time value of the router and breakdown to seconds for easy calculation of the data traffic. The normal data packet will be reached destination as the way but the DDos attacked packet will get differ from the other, this might happened by the zombies. They can consider single attack part other than the overlap packet attack. This is done in two steps: First, users are allowed to pass through router in network site in that it incorporates Detection Algorithm and detects for legitimate user. Second, again it pass through router placed in cloud site in that it incorporates confirmation Algorithm and checks for threshold value, if it ’s beyond the threshold value it considered as legitimate user, else it’s an intruder found in environment. This System is represented and maintained by as third party. When attack happens in environment, it sends notification message for client and advisory report to Cloud Service Provider (CSP).

ISSN 2348 – 7968

B. Authenticate Cloud Server

module contains both the administrator and user authentications. The admin would have the privilege to view the whole process by the user. Once the user registers, user would be able to view only the authenticated page. The personal information and the data which are transferred by the user can be viewed by the user. Once logged in the server would be able to receive the data packets. The network is classified by workgroups. The active and the connected systems over the network are obtained with the use of this module. Once logged in to the process, the module obtains the active systems and displays to the user. The user would be able to choose the system to which the data needs to be transmitted by file transfer.

C. VM Profiling

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, etc. One major factor that counts towards a VM profile is its connectivity with other VMs. Also require the knowledge of services running on a VM so as to verify the authenticity of alerts pertaining to that VM. An attacker can use port scanning program to perform an intense examination of the network to look for open ports on any VM. So information about any open ports on a VM and the history of opened ports plays a significant role in determining vulnerable in the VM. All these factors combined will form the VM profile.

VM profiles are maintained in a database and contain comprehensive information about vulnerabilities, alert and traffic. The data comes from Attack graph generator, while generating the attack graph, every detected vulnerability is added to its corresponding VM entry in the database. According to the Decentralized Network controller the VM profile database consist the traffic patterns involving the VM are based on 5 tuples (source MAC address, destination MAC address, source IP address, destination IP address, and protocol). We can have traffic pattern where packets from a single IP and are delivered to multiple destination IP addresses, and vice-versa. we assume that an attacker can be located either outside or inside of the virtual networking system. The attacker's primary goal is to exploit vulnerable VMs and compromise them as zombies.

Our protection model focuses on virtual-network-based attack detection and reconfiguration solutions to improve the resiliency to zombie explorations. Cloud Service Provider (CSP) is free to install whatever operating systems or applications. The client data which are transferred to the destination need to be monitored. Reports are of no use after the data has been affected by the intruder. Each data information path is traced back from one end to the other. When the data gets deviated from the desired path, the monitoring stub would report to the client.

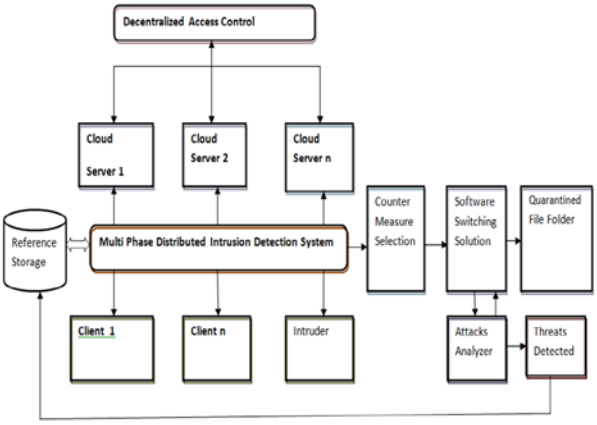

Fig.1 :.System Architecture

D. Network Intrusion Detection and Prevention System

In DNIDPS, we propose a High Performance Hash-Based string Matching Algorithm for NIPS which algorithm that places the set of the attacks signatures in the TCAM. It is capable of matching multiple patterns and attains line rate speed and greater accuracy of detection is comparatively easy to analyse and implement an efficient TCAM based implementation of Multi Pattern Matching using covered state encoding uses a state encoding scheme called a covered state encoding for the efficient TCAM-based implementation of the Aho-Corasick multi pattern matching algorithm, which is used in network intrusion detection systems.

It also proposes constructing the modified Aho-Corasick NFA for multi character processing, which can be implemented on a TCAM using the covered state encoding. The covered state encoding takes advantage of "don't care" feature of TCAMs and information of failure transitions is implicitly captured in the covered state encoding. The algorithm follows a string of width 'w' bytes is fetched, the string's rightmost block 'b' bytes are inserted as a key to the shift table and the shift is retrieved. If the shift value N is not equal to 0, the text position is shifted right by N. If it is 0, then possible pattern match is found and it needs to look at the patterns pointers. It follows the hash entry's list, which points to the patterns that have the same key as the matched string.It maps a single character to a shift value; create shift values for a block which contains B characters. Using Blocks reduce the amount of false matches. When a packet is processed, a block of characters is extracted from the search window (right to left) which is used as an index to the shift table. The shift value is retrieved from the shift table. If it is zero, then an exact match must be performed, otherwise we can shift the text.

ISSN 2348 – 7968

In order to minimize the search space, the patterns are linked using a hash value. Then calculate a hash value using the sliding window's text and locate the patterns bucket in the hash-table. This paper proposes an alternative algorithm using a Hash Function which uses a SRAM that creates fingerprints of the packet payload which are then compared with the patterns signatures

Data Structures that are used by Hash-based Pattern matching Algorithm.

1. Shift Table - A shift table for a block of characters B as in the algorithm. The block of characters is used during preprocessing the patterns to construct the shifting table. Within the sliding widow, it looks at the text, B characters at a time. For simplicity assume that the table size is SB, where S is the alphabet. Each entry corresponds to a distinct substring of length B. Let X = {x1,x2, ..,xB} be a string corresponding to the i_{th} entry of the shift table. There are two cases: either X appear somewhere in one of the patterns or not. If X does not appear in any of the patterns, it stores m−B+1 in the corresponding shift entry, otherwise, it finds the rightmost occurrence of X in any of the patterns that contain it; suppose it is in P_j and that X ends at position q of P_j. Then it stores m−q in the table. If the shift value is greater than zero, it can safely shift. Otherwise, it is possible that the current substring we are looking at in the text matches some pattern in the pattern list. To avoid comparing the substring to every pattern in the pattern list, it uses the previous defined hash table (that minimizes the number of patterns to be compared).

2. Patterns Table - an array of patterns ordered by a pattern ID.

3. Matched Patterns List - each entry contains the matched patterns and its corresponding end position in the text.

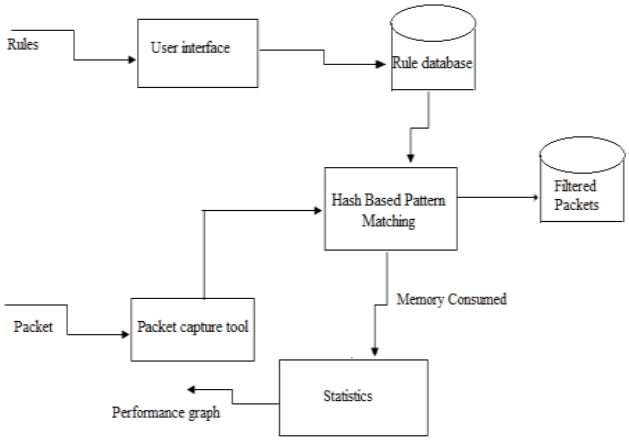

Fig 2 : Hash Based Pattern Matching system

Algorithm : Hash-Based Pattern Matching

1: T(Packet) = {Ti, 1 ≤ i ≤ n}
2: pos⇐ 1;
3: shift ⇐ 0
4: width, w⇐ default
5: Block, b⇐ default
6: while pos ≤ n − width do
7: block⇐ $T_{[pos+width-B,..,pos+width-1]}$
8: shift ⇐ SHIFT TABLE[block]
9: if shift is 0 then
10: key⇐ hash($T_{[pos,..,pos+width-1]}$) {construct a fingerprint}
11: Step (2)
12: entry= SRAM.lookup(key)
13: Step (3)
14: shift □ entry.shift
15: if shift ≠ 0 then
16: pos□pos+shift
17: continue
18: end if
19: Step (6)
20: for all current=entry.key.next ≠ null do
21:if current.len ≤ width {exact match} OR checkSubPatterns(current.len ,pos, current.SRAM Ptrs) = = True then
22: MatchedList.add(current.Id, pos+current.Len)
23: end if
24: end for
25: end

E. Attack Analyzer

An attack graph is a modeling tool to illustrate all possible multi-stage, multi-host attack paths that are crucial to understand threats and then to decide appropriate countermeasures. In an attack graph, each node represents either precondition or consequence of an exploit. Attack graph is helpful in identifying potential threats, possible attacks and known vulnerabilities in a cloud system.

Definition 1 (Scenario Attack Graph). An Scenario Attack Graph is a tuple SAG= (V, E), where,

1. V = NC[ND[NR denotes a set of vertices that include three types namely conjunction node NC to represent exploit, disjunction node ND to denote result of exploit, and root node NR for showing initial step of an attack scenario.

2. E = Epre [Epost denotes the set of directed edges. An edge e 2 Epre _ ND _ NC represents that ND must be satisfied to achieve NC. An edge e 2 Epost _ NC _ ND means that the consequence shown by ND can be obtained if NC is satisfied.

Node VC E NC is defined as a three tuple (Hosts; vulalert) representing a set of IP addresses, vulnerability

ISSN 2348 – 7968

information such as CVE, and alerts related to vc, respectively. ND behaves like a logical OR operation and contains details of the results of actions.

- NR represents the root node of the scenario attack graph. For correlating the alerts, we refer the define a new Alert Correlation Graph (ACG) to map alerts in ACG to their respective nodes in SAG. To keep track of attack progress, we track the source and destination IP addresses for attack activities.

Definition 2 (Alert Correlation Graph). An ACG is a three tuple ACG = (A;E; P), where A is a set of aggregated alerts. An alert a 2 A is a data structure (src; dst; cls; ts) representing source IP address, destination IP address, type of the alert, and timestamp of the alert respectively.

- Each alert a maps to a pair of vertices (vc; vd) in SAG using map (a) function, E is a set of directed edges representing correlation between two alerts

- P is set of paths in ACG.

ALGORITHM : ALERT CORRELATION

Here is a method for utilizing SAG and ACG together so as to forecast an attacker's behavior. Alert Correlation algorithm is implemented for every alert detected and returns one or more paths Si. For every alert ac that is received from the IDS, it is added to ACG if it does not stay alive.

Require: alert ac, SAG, ACG
1: if (ac is a new alert) then
2: create node ac in ACG
3: n1 vc € map (ac)
4: for all n2 € parent (n1) do
5: create edge (n2.alert, ac)
6: for all Si containing a do
7: if a is the last element in Si then
8: append ac to Si
9: else
10: create path Si+1 = {subset(S I,a), ac}
11: end if
12: end for
13: add ac to n1.alert
14: end for
15: end if
16: return S

Definition 3 (VM State). Based on the information gathered from the decentralized network controller, VM states can be defined as following:

- Stable: there does not exist any known vulnerability on the VM.
- Vulnerable: presence of one or more vulnerabilities on a VM, which remains unexploited.
- Exploited: at least one vulnerability has been exploited and the VM is compromised.
- Zombie: VM is under control of attacker

F. COUNTERMEASURE SELECTION

we present the methods for selecting the countermeasures for a given attack scenario. The countermeasure serves the purpose of 1) protecting the target VMs from being compromised; and 2) making attack behavior stand prominent so that the attackers' actions can be identified

Definition 4 (Countermeasure Pool).
A countermeasure pool CM = (cm1; cm2;: : ; cmn) is a set of countermeasures. Where
1. Cost is the unit that describes the expenses required to apply the countermeasure in terms of resources and operational complexity, and it is defined in a range from 1 to 5, and higher metric means higher cost;
2. intrusiveness is the negative effect that a countermeasure brings to the Service Level Agreement (SLA) and its value ranges from the least intrusive (1) to the most intrusive (5), and the value of intrusiveness is 0 if the countermeasure has no impacts on the SLA;
3.Condition is the requirement for the corresponding countermeasure;
4. Effectiveness is the percentage of probability changes of the node, for which this countermeasure is applied.

Finally, SAG and ACG are also reorganized before terminating the algorithm. The complexity of Algorithm 2 is O(|V| * |CM|), where |v| is the number of vulnerabilities.

Countermeasure selection Algorithm

Require: Alert, G(E,V), CM
1: Let valert= Source node of the Alert
2: if Distance to Target(valert) > threshold then
3: Update_ACG
4: return
5: end if
6: Let T = Descendant(valert) U valert
7: Set pr(valert)=1
8: Calculate_Risk_Prob(T)
9: Let benefit[|T|,|CM|]=0
10: for each t € T do
11: for each cm € CM do
12: if cm.condition(t) then
13: Pr(t) = Pr(t)*(1-cm.effectiveness)
14: Calculate_Risk_Prob(Descendant(t))
15: benefit[t,cm] =ΔPr(target_node)
16: end if
17: end for

ISSN 2348 – 7968

18: end for
19: Let ROI[|T|,|CM|]=0
20: for each t €T do
21: for each cm € CM do
22: ROI[t,cm]= benefit[t,cm]/cost.cm+intrusiveness.cm
23: end for
24: end for
25: Update SAG and Update ACG
26: return Select Optimal CM (ROI)

IV.CONCLUSION

DNIDPS utilizes the attack graph model to carry out attack detection and prediction. The system evaluate and demonstrates the feasibility of Network Intrusion Detection and Prevention System, that the proposed solution can significantly diminish the risk of the cloud system from being exploited and abused by internal and external attackers. DNIDPS only investigates the network IDPS approach to counter DDOS attacks. Additionally, the scalability of the proposed DNIDPS solution by investigating the decentralized network control and attack analysis model

References

[1] .Carlo Marcelo Revoredo da Silva, Jose Lutiano Costa da Silva,Ricardo Batista Rodrigues, Leandro Marques do Nascimento,Vinicius Cardaso Garcia, “Systematic Mapping Study on Security threats in Cloud Computing”. (IJCSIS) International Journal of Computer Science and Information Security, Vol. 11, No. 3, March 2013.
[2]. Martin R. Stytz, Ph.D. Sheila B. Banks, Ph.D., “Cyber Warfare Simulation to prepare to control cyber space”.AnirudhRamachandran and Nick Feamster, College of Computing, Georgia Tech, “Understanding the Network-Level Behavior of Spammers”.
[3]. Fernando Sanchez, Zhenhai Duan, Yingfei Dong , “Understanding Forgery Properties Of Spam Delivery Paths”.
[4]. Jiri Matas, Jan Sochman, “Wald’s Sequential Analysis For Timeconstrained Vision Problems” , Springer US, Unifying Perspectives in Computational and Robot Vision Volume 8, 2008, pp 57-77.
[5]. Sang Yun, “An Efficient TCAM Based Implementation Of Multi Pattern Matching Using Covered State Encoding", vol.62(2), pp.213-221, February 2012.
[6]. R. M. Karp and M. O. Rabin, "Efficient Randomized Pattern Matching Algorithms", Technical report TR-31-81, Harvard University, Cambridge, MA, USA, December 1981.
[7]. S. Wu and U. Manber, "Fast Text Searching with Errors", Technical Report TR-91-11, University of Arizona, Department of Computer Science, June 1991.
[8]. S. Wu and U. Manber. Agrep , "A Fast Approximate Pattern-Matching Tool", In Proceedings USENIX Winter 1992 Technical Conference, pages 153–162, San Francisco, CA, January 1992.
[9]. S. Wu and U. Manber, "A fast Algorithm for Multi-Pattern Searching ", Technical Report TR-94-17, Department of Computer Science.