International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

ENHANCED DYNAMIC PROTECTION SCHEME IN SAAS IN CLOUDS USING ANOMALY SOFTWARE AGENT SYSTEM

B.Prakash

(Third Year MCA, Nandha Engineering College, Erode, Anna University, Tanil Nadu)
mcaprakashb@gmail.com

***Abstract*— SaaS cloud systems often host long-running applications like massive data processing, which provides more opportunities for attackers to exploit the system vulnerability and leak the information to misuse. In this paper we propose an enhanced Dynamic security scheme in SaaS in Clouds using Anomaly Software Agent system. The primary benefit of an Agent-based Information Leakage Detection system lies in the ability to modify and add detection capabilities, modularize those capabilities, and then conditionally employ such capabilities at the discretion of a central control mechanism (in our system, the Controller Agent). The use of mobile agents as described in this paper, and in general, reduces the per-host administrative complexity as once the initial agent environment is properly installed and configured; all further necessary actions are performed by the agents themselves. Additionally, mobile agents are able to provide unique reporting capabilities that, for the purposes of our research, may benefit the analysis of information leakage, protection and the underlying covert channels through which information has been leaked.**

***Index Terms*—Distributed Service, Data Privacy, Application Service Providers (ASPs), Anomaly Software Agent.** *(Key words)*

I. INTRODUCTION

Cloud computing is a technology helps us to keep up data and its application by using internet and central remote servers [3]. Cloud computing has greater flexibility and availability at lower cost. The four deployment models operated by cloud computing are the: Public Cloud, Private Cloud, Community Cloud, and Hybrid Cloud. Private cloud -- The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise. Community cloud -- The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns. It may be managed by the organizations or a third party and may exist on premise or off premise. There are different types of cloud service providers like Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).Here we are discussing about how to protect leakage in SaaS Cloud server.

The Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made this is available to customers over a network. SaaS service are suffered from many malicious attacks hence they need security.

We propose an information leakage detection (ILD) agent system to automate the processes of converting a regular cloud server to colored one.(i.e. SaaS cloud server) Furthermore, The distributed reporting potential of mobile agent networks can lend itself well to future analysis of information leakage, as well as the underlying covert channel techniques. The agent based approach also makes the coloring scheme effective in an open system which is a hybrid of machines running modified cloud systems and commodity ones. Given comparable requirements for a small memory footprint and ease of integration with relatively low-level system constructs necessary to accomplish efficient file system monitoring process.

II. RELATED WORK

The previous work has provided various software integrity attestation solutions [1], [2], [3], [4]-[8], those techniques often require special trusted hardware or secure kernel support, which makes them difficult to be deployed on large-scale cloud computing infrastructures. Traditional Byzantine fault tolerance (BFT) techniques [9], [10] can detect arbitrary misbehaviors using full-time majority voting (FTMV) over all replicas, which however incur high overhead to the cloud system.

In this section, we present IntTest, a new integrated service integrity attestation framework for multitenant cloud systems. IntTest provides a practical service integrity attestation scheme that does not assume trusted entities on third-party service provisioning sites or require application modifications.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

IntTest builds upon our previous work RunTest [10] and AdapTest [11] but can provide stronger malicious attacker pinpointing power than RunTest and AdapTest. Specifically, both RunText and AdapTest as well as traditional majority voting schemes need to assume that benign service providers take majority in every service function. However, in large-scale multitenant cloud systems, multiple malicious attackers may launch colluding attacks on certain targeted service functions to invalidate the assumption. To address the challenge, IntTest takes a holistic approach by systematically examining both consistency and inconsistency relationships among different service providers within the entire cloud system. IntTest examines both per-function consistency graphs and the global inconsistency graph.

The per-function consistency graph analysis can limit the scope of damage caused by colluding attackers, while the global inconsistency graph analysis can effectively expose those attackers that try to compromise many service functions. Hence, IntTest can still pinpoint malicious attackers even if they become majority for some service functions.

III. PROBLEM FORMATION

Given an SaaS cloud system, the goal of ILD agent system is to pinpoint any malicious service provider that offers an untruthful service function. ILD agent system treats all service components as black boxes, which does not require any special hardware or secure kernel support on the cloud platform. The automate the process of detecting and coloring receptive hosts' file systems and monitoring the colored file system for instances of potential information leakage.

IV. ILD AGENT SYSTEM

Separation of powers and responsibilities in an agent community encourages flexibility and encapsulation. As such, our proposed agent system will be heterogeneous with members belonging to one of six principle archetypes, each adhering to unique roles and possessing distinct abilities. Figure 1 depicts the classifications of our Information Leakage Detection (ILD) Agent system and the respective agent ranks. All inter-agent communications will adhere to FIPA Agent Communication Language (ACL) specifications in order to maintain communication interoperability between different agent platforms, Properties and responsibilities of each type of agent are discussed in following subsections.

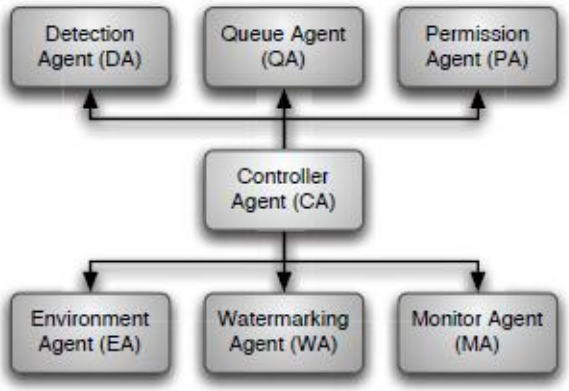

Fig. 1. Agent Classifications and Hierarchy

A. Controller Agents (CA)

Controller Agents are responsible for dispatching subordinate agents and coordinating their respective activities in a designated network. Additionally, Controller Agents will coordinate the remote installation of the necessary mobile agent environment and other required software packages on target hosts with Environment Agents. Multiple instances of controller agents can be dispatched to ensure proper coverage of large networks as well as to accomplish load distribution for the purposes of performance optimization.

B. Detection Agents (DA)

The main functionality of Detection Agents is to identify new hosts in the network and to verify the host's states. In our initial design, a host's state will refer to the presence or absence of untrusted cloud server and the trusted cloud server infrastructure. Once determined, a host's state will be reported to the Controller Agent to aid in the identification of subsequent actions.

C. Queue Agents (QA)

To avoid overwhelming Controller Agents and to provide an orderly approach to dispatching agents to newly discovered hosts, Queue Agents will be useful. As stated above, when a Detection Agent identifies a new remote host, the host's state is reported to a Controller Agent. Rather than dispatching agents to a new host immediately, it may be preferred to defer such processing for some time, especially in the case when many such hosts are reported at once. In such cases, hosts are reported by Controller Agents to Queue Agents which

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

prioritize hosts for subsequent processing by, and at the request of, Controller Agents.

D. Monitor Agents (MA)

Monitor Agents will perform active monitoring on the host file system through the subsystem to identify file write and creation operations. Details on the subsystem will be discussed in the next section. When a write operation or file creation operation takes place, Monitor Agents notify Watermarking Agents which can then perform watermark analysis of the file in question.

As comparable capabilities are already present in trusted cloud server hosts, Monitor Agents will only reside in untrusted cloud server host machine.

E. Watermarking Agents (WA)

Similar to Monitor Agents, Watermarking Agents shall only be present untrusted (become malicious) cloud server as determined by Detection Agents. The responsibility of these agents is to watermark all files on a host's file system and to perform subsequent watermark analysis at the request of Monitor Agents.

F. Permission Agents (PA)

A central Permission Agent handles permissions issues involving Monitor Agents and Watermarking Agents with their target hosts. Specifically, the Permission Agent should ensure that such agents are given only those permissions necessary to perform their respective tasks. In addition, the Permission Agent ensures that all permissions necessary for agent environment installation by the Environment Agent are in place.

G. Environment Agents (EA)

Minimally, Watermarking and Monitor Agents require the necessary agent environment installed on a target host in order to reside and function there. Also, depending on the type of watermarking employed, certain watermarking specific software dependencies which may not reasonably be accommodated by the Watermarking Agents themselves can exist. Environment Agents will be responsible for handling all such software dependencies without the intervention of the target host's administrator.

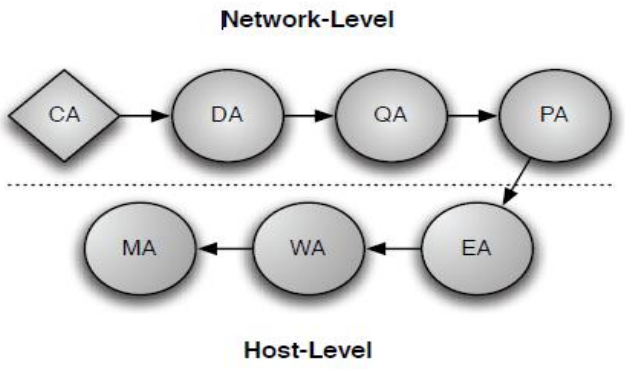

Fig. 2. Process flow of our proposed system.

V. PROPOSED STRATEGIES

A. Host Discovery

In our proposed agent system, all operations begin with, and are coordinated by, the Controller Agent. Initially, it is assumed that all hosts in the network are clean, yet unknown.

A Detection Agent is dispatched to scan the network for untrusted cloud server hosts. When the first such host is discovered, the Detection Agent determines whether or not the newly found host is "Colored." If the host is trusted cloud server (benign server) means, it is reported to the Controller Agent.

B. Non-Colored Host Queuing

When the first non-Colored, untrusted cloud server -based host is identified and reported by the Detection Agent, the Controller Agent shall create a Queue Agent and make it aware of the reported host.

All subsequent host reports generated by the Detection Agent will also be forwarded to the Queue Agent. Hosts are enquired, possibly with priorities, by the Queue Agent. At certain times, the Controller Agent will query the Queue Agent for a new host, which the Queue Agent will reverse queue and forward to the Controller Agent.

C. Permission Determination and Management

Given a host report from the Queue Agent, the Controller Agent will create a Permission Agent and assign it to the new host. The permission agent will attempt to determine if the proper permissions are in place for the successful remote installation of an agent environment on the target host, and for the proper operation of subsequently dispatched Watermarking and Monitor agents.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

If proper permissions have not been assigned, the Permission Agent is responsible for coordinating with the target host to establish the lacking permissions at Once this process has completed.

The Controller Agent remotely installs (with the aid of a helper Environment Agent) the appropriate agent environment on the target host.

D. Watermarking Target Hosts

Following the successful installation of the agent environment on the target host, the Controller Agent dispatches a Watermarking Agent to the host. Within the host, the Watermarking Agent "colors" all files on the host's file system. Upon completion of initial coloring, the Watermarking Agent reports completion to the Controller Agent, and then awaits subsequent commands. Detection of a newly created file, or of write operations performed on an existing file, are reported to the Watermarking Agent by the Monitor Agent, prompting the Watermarking Agent to analyze and possibly color the new file. This process continues until the Controller Agent instructs the Watermarking Agent to terminate. This agent will then use the proposed methods to detect and handle potential instances of information leakage.

VI. IMPLEMENTATION AND RESULTS

A. Agent Environment

In choosing an appropriate foundation for our agent community, we considered primarily the associated memory footprint as well as ease of access to system-level constructs. Mobile-C was hence accepted as our mobile agent framework due to its low memory footprint when compared to other popular agent architectures. In addition, being fully compliant enables Mobile-C agents to take direct advantage of the system calls provided by the Anomaly Software Agent system. This is especially useful for our purposes as our Monitor Agent relies on Controller Agents system.

B. Watermarking Algorithms

As different file types require different watermarking schemes, we focused on image files for our experiments. The watermarking algorithm utilized is the Dugad [12] algorithm as implemented in Peter Meerwald's watermarking library. This algorithm has many nice properties, especially that of blindness, which is required for our system.

C. Handling Dependencies

External dependencies can be handled in several ways in mobile agent systems. Ideally, all necessary code can efficiently be carried with the agent itself. When this is not viable, the agent execution environment can be made to handle such dependencies. An Environment Agent capable of retrieving, building, and installing into the execution environment packages which are needed by Watermarking Agents shall be employed. This will be helpful as new watermarking techniques and information leakage detection methods are developed which may require large and complex software suites to function.

D. Implementation of the Watermarking Agent

As described above, the primary role of a Watermarking Agent is to prepare a file system for information leakage detection by watermarking all files with a particular permissions tag. Such tags essentially identify the sensitivity of a file and are used in conjunction with permissions assigned to individual users. A user's permissions regulate which files are accessible by the user. Here, accessibility can relate to the ability of a user to read, write, or execute a file, or perform any combination of these actions. Information leakage via covert channels may result in the removal or modification of traditional permissions tags. The recipient of the leaked information may alter the tags in order to grant himself access to the information that he was not intended to possess. Functionally, the Watermarking Agent developed for our experiments initiates a complete scan of the target file system upon entry into a target host. It could be the case that the file system, or portions of it, is already watermarked but the agent, agency, or supporting infrastructure was damaged or removed due to some unforeseen circumstance.

Therefore, the Watermarking Agent will attempt to detect the presence of a watermark in all scanned files prior to watermarking. If a watermark is not detected, the file is watermarked immediately with a signature corresponding to the files permissions tag. Conversely, if a watermark is detected, the Watermarking Agent will compare the watermark with the file's permissions tag. If an inconsistency is found, the file is assumed to have been previously leaked, and is either quarantined in a secure directory or securely deleted. Once the initial watermarking phase is complete, the Watermarking Agent will become dormant. A Watermarking Agent will be awakened upon receipt of signal from the Monitor Agent indicating that a new file has been created and will therefore need to be watermarked1. ***Algorithm 1*** provides a broad representation of the operations performed by our Watermarking Agent.

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)
http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

Algorithm 1 *Watermark (Directory D)*

```
1: while D has children do
2:    di child i of D
3:    if di is a directory then
4:       Watermark(di)
5:    else
6:       boolean w = DetectWatermark(di)
7:    if w = TRUE then
8:       Compare watermark of di with
         permissions tag
9:       if Watermark does not match tag then
10:         Quarantine or Securely Remove di
11:      end if
12:   else
13:      Watermark di with signature =
         permissions tag
14:   end if
15: end if
16:end while
17: return
```

E. Implementation of the Monitor Agent

While the Watermarking Agent effectively binds a files permissions tag to its content, it does not compare the watermark to the permissions of a user attempting to access the file. This task is the responsibility of the Monitor Agent. The Monitor Agent serves the primary role of monitoring the target file system for any file "creation" or "write" operations and notifying the watermarking Agent of such events for subsequent processing. As stated above, ***Algorithm 2*** steps represent the Monitor Agent operations.

Algorithm 2 Monitor()

```
1: W ← inotify event descriptor
2: for all Target directories di do
3:    Add inotify watch descriptor for "write"  and "create"
      operations within di
4: end for
5: loop
6:    f ← Read event from event descriptor W
7:    Pass f to Watermarking Agent for Analysis
8: end loop
```

F. Results

Regardless of the type of covert channel through which information is leaked, the detection methods of effectively prevent any disassociation of the leaked information content from its designated permissions from being used by the recipient of the leaked information. If permissions 1For future works, the Watermarking Agent shall be made able to detect valid changes of permissions tags, and re-watermark files accordingly.

VII.CONCLUSION AND FUTUTRE WORK

In this paper ,we have presented ILD system (i.e. Software Agent system) to automate the process of detecting and coloring receptive hosts' file systems and monitoring the colored file system for instances of potential information leakage in SaaS clouds. Furthermore, ILD system provides result auto correction to automatically correct compromised results to improve the result quality. Our experimental results show that it can achieve higher leakage protection accuracy than existing alternative schemes. Agent systems are lightweight, which imposes low-performance impact to the data processing services running inside the cloud infrastructure.

Future work in this area may lead to the inclusion of techniques aimed at detecting and blocking covert channels prior to the occurrence of information leakage. Given the highly varied nature of covert channeling methods, detecting all such methods is likely a matter for which a solution can only be obtained through the liberal use of techniques rooted deeply in the field of cloud security.

VIII.REFERENCES

[1] J. Garay and L. Huelsbergen, "Software Integrity Protection Using Timed Executable gents,"Proc. Mar. 2006

[2] S. Berger et al., "TVDc: Managing Security in the Trusted Virtual Datacenter," ACM IGOPS Operating Systems Rev., vol. 42, no. 1, pp. 40-47, 2008.

[3] T. Garfinkel et al., "Terra: A Virtual Machine-Based Platform for Trusted Computing,"Proc.19th ACM Symp. Operating Systems Principles (SOSP), Oct. 2003.

[4] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla,"Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems," Proc. 20th ACM Symp.Oct. 2005.

[5] E. Shi, A. Perrig, and L.V. Doorn, "Bind: A Fine-Grained Attestation Service for Secure Distributed Systems," Proc. IEEE Symp. Security and Privacy, 2005.

International Journal of Computer Science
ISSN:2348-6600
Oddity...Probe...Reviste...
Published by SK Research Group of Companies(SKRGC)

http://www.ijcsjournal.com Volume 3, Issue 1, No 5, 2015. ISSN: 2348-6600
Reference ID: IJCS-092 PAGE NO: 525-530.

[6] The Trusted Computing Group website, https://www.trustedcomputinggroup.org, 2013.

[7] J.L. Griffin, T. Jaeger, R. Perez, and R. Sailer, "Trusted Virtual Domains: Toward Secure Distributed Services," Proc. First Workshop Hot Topics in System Dependability, June 2005.

[8] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Trans.Programming Languages and Systems, vol. 4,no. 3, pp. 382-401, 1982.

[9] T. Ho et al., "Byzantine Modification Detection in Multicast Networks Using Randomized Network Coding," Proc. IEEE Int'l Symp. Information Theory (ISIT), 2004.

[10] J.Du,W.Wei, X. Gu, and T. Yu, "Runtest: Assuring Integrity of Dataflow Processing in Cloud Computing Infrastructures,"ACM Symp. (ASIACCS), 2010.

[11] J. Du, N. Shah, and X. Gu, "Adaptive Data-Driven Service Integrity Attestation for Multi-Tenant Clo`ud Systems," Proc. Int'l Workshop Quality of Service (IWQoS), 2011.

[12]R. Dugad, K. Ratakonda, and N. Ahuja, "A New Wavelet-based Scheme for Watermarking Images". In Proceedings of the International Conference on Image Processing, vol. 2, pp. 419-423, Oct. 1998.