RESEARCH ARTICLE OPEN ACCESS

Preventing Covert Channel Attacks By Using Software Agents

Deepika Sonal[1], D.Kiranmayi[2]
[1](Department Of Computer Science and Engineering, Vignan Institute of Information Technology, Visakhapatnam, Andhra Pradesh.)
[2] (Department Of Computer Science and Engineering, Vignan Institute of Information Technology, Visakhapatnam, Andhra Pradesh)

Abstract*:*
Information sharing and their protection is a major problem for organisations having sensitive data. Shared resources are utilized by the covert channel for indirectly transmit sensitive information to unauthorized parties**.** For security purpose some mechanism such as seLINUX rely on the tagging the file system with access control properties. Such mechanism does not provide strong protection so colored LINUX, an extension to seLINUX is used to "color" the content of the file according to security classification to enhance resistance information laundering. In this paper we discuss about the mobile agent based approach to provide the environment to automate the process of detecting and coloring the respective file system(host) and monitoring the colored file system for instance of potential information leakage.
Keywords: covert channel, information laundering, information leakage, notify, overwhelming

I. Introduction

Information protection is one of the most pressing current challenges in distributed system. Different organization has sensitive information, which needs to be transferred by covert channel, e.g. personal health information, credit card details within banks, government agencies and so on. A covert channel is a by product of shared resources like memory, network, and execution time on computing device and can be created and accessed dynamically [1]. Because the covert channels are created from shared resources, it is difficult to detect and prevent their occurrences**.** Covert channel attacks utilise shared resources to indirectly transmit sensitive information to unauthorised parties.

According to survey in 2006 of global security system, 28 percent of information leakage contributed 18 percent of internal breaches [2]. Internal breaches occurs inside the organisations by the legitimate and authenticated users, most conventional security measurements cannot effectively detect and prevent such activities. Now days operating systems counter unauthorized accesses through the use of access control tags, or labels applied to subjects (processes or users) and objects (files . Now these labels are compared with the permissions assigned to users attempting to access the labelled files. SELINUX is use to provide this mechanism, which is effective in access control in most of the situation, it is vulnerable to covert channel attacks. The attacks enable laundering of access control tags or tags reassignment. An extension to SELINUX that is colored LINUX [3] provides data watermarking, or coloring. The main advantage of their approach is that it does not need any knowledge of covert channels since it modification of operating system is on filesystem kernel to monitor read and write accesses. But this is strongly applicable for "close" system that has modified operating system.

To overcome this drawback, we proposed an information leakage detection (ILD) agent system. This approach involves the ability to modify and add detection capability in modular fashion and also provide conditional deployment of such capabilities with the help of mobile agent. The agent based approach also makes the coloring scheme effective in an open system which is a hybrid of machines running modified operating system. Mobile-c [4] is chosen for mobile agent platform, as it meets all our requirements.

In our next section 2, related work on information detection is presented. Section 3 details about agent system, section 4 and 5 provides detection schemes and proposed strategies, section 6 communication systems present in agent model, section 7 and 8 contains implementation details and results. Finally section 9 provides conclusion and future directions.

II. Related work

Most of the work done in network security is for preventing outsider access, only few literatures are base on insider leakage of information's. To protect share information among organization one mechanism is introduce that is Trusted Platform Module (TPM) [5]. TPM devices introduce that data can be shared only by trusted devices. Cover channel attacks are discussed on [6]. covert channel on

network based storage based on IP to live (TTL) is designed in [7]. A link layer network based covert channel in MAC protocol based on the splitting algorithm is proposed in [8]. Detection of covert channel attacks so many methods are proposed such as information flow analysis technique [9], time-domain anomaly, but it is impossible to enumerate all covert channels in a real system.
Watermarking offers a strong security tags and can be detect information leakage from both insiders as well as through covert channels attacks. Finally, approach proposed in this paper to colored LINUX methodologies with mobile agent is novel.

III. Agent system

Classification of information leakage detection (ILD) agent system is given in the figure 1. Properties and responsibilities of each type of agent are differ from each other and discussed below.

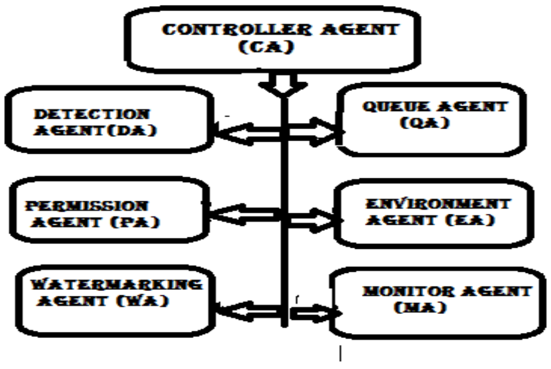

Figure 1. Agent classification.

1. **Controller agent (CA)**
   CA is responsible for coordinating activities and dispatching to subordinate agents.
2. **Detection agent(DA)**
   DA is to identify new host in the network and to verify the host's states.
3. **Queue agent**
   QA provides the ordering approach to dispatching agents to newly discovered hosts. QA avoid the overwhelming of agents.
4. **Monitoring agent (QA)**
   MA will perform active monitoring on the host filesystem through the inotify kernel subsystem to identify file write and creation operation.
5. **Watermarking agent(WA)**
   The responsibility of this agent is to watermark all files on the host's filesystem and to perform subsequent watermark analysis at the request of Monitor agents.
6. **Permission agents(PA)**
   PA handles permissions issues involving Monitor agent and Watermarking agents with their target host.
7. **Environment agents(EA)**
   EA is responsible for handling necessary agent environment such as software dependencies without the intervention of the target host's administrator.

IV. Proposed strategies

Process flow is depicted in figure 2.

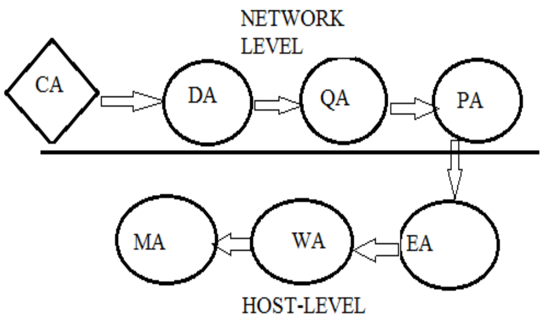

Fig. 2. process flow

Objective of each step and how they can achieved explains in below subsections

1. Host in the network
In this agent system all operations begins with, and are coordinated by CA. Initially networks are clean, yet unknown. A DA is dispatched to scan the network for SELINUX-based hosts. After discovering the first host the DA is to determine whether or not the newly host is colored. If the host is un-colored, it is reported to the CA.
2**. Non-colored host**
Now the CA will create a queue agent and aware it with the reported host. After that all the host reports will also be forwarded to the QA. And when CA will query for the new host, than the QA will dequeue and forward to CA.
3. **Permission granted and management**
CA assigns a new host to PA. PA use standard LINUX remote management facilities, as a mobile agent environment has not yet been installed on the target host and will attempt to determine proper permission to target host and operation of subsequently dispatched. Once this process has completed, the CA remotely installs the appropriate agent environment on the target host.
4. Watermarking target host
After installation of the agent environment on the target host, the CA dispatches WA to the host, WA "colored" all files on the host's filesystem. This process continues until the controller agent instructs the Watermarking Agent to terminate.
5. Detection methodologies
Kernel –level system call hooking is use to maintain the high performance and mitigate the time cost associated with "write" and "create" operation detection, possibility and feasibility of detecting such

operations via their respective system calls at the kernel level.

V. Agent communication system

Communication among agent will follow FIPA communicative act specification which is based on speech act theory to facilitate communication interoperability between different agent platform [10]. Table 1 through 7 illustrates the communication details.

TABLE1: Controller Agent Communications

From: Controller Agent (CA)
To: Detection Agent (DA) ➢ Ask the DA to report CA when the first un-colored host is Detected. (REQUEST-WHEN) ➢ After first host found, ask the DA to report to QA whenever un-colored hosts are found. (REQUEST-WHENEVER)
To: Queue Agent (QA) ➢ Ask the QA to upgrade current non-colored hosts to its queue. (REQUEST) ➢ Retrieve the hosts in the QA's queue. (REQUEST with INFORM)
To: Permission Agent (PA) ➢ Request PA to prepare target host for agent environment installation. (REQUEST)
To: Watermarking Agent (WA) ➢ Ask the WA to watermark the host's filesystem and report the completion. (REQUEST-WHEN)
To: Monitor Agent (MA) ➢ Ask the MA to monitor the target host and notify the CA when information leakage occurred. (SUBSCRIBE)
To: Environment Agent (EA) ➢ Ask the EA to check for, and resolve, software dependencies on the target host which may inhibit the functionality of subsequently dispatched agents. (REQUEST)

TABLE 2: Detection agent communications

From: Detection Agent (DA)
To: Controller Agent (CA) ➢ Confirm to CA that network scan to determine non-colored host is proceeding. (AGREE) ➢ Report to CA when the first non-colored host is found. (INFORM) ➢ Confirm to CA that notification to QA about non-colored hosts can proceed. (AGREE)
To: Queue Agent (QA) ➢ Ask the QA to insert current non-colored hosts in its queue.

TABLE 3: Queue agent communications

From: Queue Agent (QA)
To: Controller Agent (CA) ➢ Confirm to CA that queue insertion has been performed. (AGREE) ➢ Return the current hosts in queue to CA. (INFORM)
To: Detection Agent (DA) ➢ Confirm to DA that queue insertion has occurred. (AGREE)

TABLE 4: Permission agent communications

From: Permission Agent (PA)
To: Controller Agent (CA) ➢ Confirm to CA to prepare the host for agent environment Installation. (AGREE) ➢ Notify CA of the result of host preparation. (INFORM)

TABLE 5: watermarking agent communications

From: Watermarking Agent (WA)
To: Controller Agent (CA) ➢ Confirm with CA to perform watermarking operation. (AGREE) ➢ Return the result of watermarking operation to CA. (INFORM)

TABLE 6: Monitor agent communications

From: Monitor Agent (MA
To: Controller Agent (CA)   ➢ Confirm with CA to perform queue insertion. (AGREE)   ➢ Notify CA of the occurrence of information leakage. (INFORM)

TABLE 7: Environment agent communications

From: Environment Agent (EA)   To: Controller Agent (CA)   ➢ Confirm with CA to perform environment checking and dependency Resolution. (AGREE)

VI. Implementation approach

Mobile–c is accepted as mobile agent framework due to its low memory footprint when compare to other popular agent architecture. Mobile-c agents were developed to perform initial watermarking of a portion of filesystem and also use for detection of leakage.

We focused on the image file for our experiments. The watermarking algorithm is explain in [11].this algorithm has many nice properties, especially that of blindness, which is require for system.

Dependencies can be handling by agent execution environment; mobile-c uses Ch, an embeddable, C99-compliant,c-language interpreter as its execution environment. Certain dependencies such as NetPBM, cannot be detect by the agent themselves and therefore we added several package to Ch execution environment, for this purpose watermarking agents shall be employed.

Watermarking agents are prepared by watermarking all files with particular permission tags in filesystem. Given tags are essentially identify the sensitivity of a file are used in conjunction assigned to individual user's. Watermarking agents will detect the presence of watermark in all scanned file prior to watermarking. If it is not detected then it provides it immediately. And in the case of detection watermark WA will compare the watermark with the permission tags. Algorithm 1 provides the work done by watermarking agent.

Monitor agents perform the primary role of monitoring the target filesystem for any "creation" and "write" operations and notifying the watermarked agent for further processing. Algorithm 2 explain the performance of the monitoring agent operation in notify kernel system.

Algorithm 1

Watermark (file C)

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

Algorithm 2

Monitor()

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

VII. Results

In the covert channel, if the information is leaked than the detection methods prevents any disassociation of leaked information. If the permission is changed or altered during leakage, they will not match the information's embedded watermark. And if the information is altered, then the watermark will no longer be valid. Initial test is conducted on Intel-based machine with LINUX kernel version 2.6 and SELINUX security extension. Watermarking test was introducing in test and next is monitor agent was introduced that tested the functionality by creating new file, and writing the existing files in the filesystem.

Agents correctly identified the invalid watermarks by comparing it with permission tags and modified it. Information leakage detection is vibrant and further researches needs for continue development in this field.

VIII. Conclusion and future work

Leakage detection approach on agent system based modifies and adds the capability of central control mechanism in distributed system. All the actions are performing by the different agents mention in the paper. Mobile agents are able to provide the unique detection scheme that will benefit the detection and control of information leakage in covert channel.

Future work should done in this area is to provide blockage in covert channel if prior information of leakage is detected. Covert channels sense the attacks before the transfer of the information by providing some basic sense of detection from prior attacks and update it with some kind of intelligence.

REFERENCES

[1.] National Computer Security Center. "A Guide to Understanding Covert Channel Analysis of Trusted Systems" NCSC-TG-30, November 1993, http://www.radium.ncsc.mil/tpep/library/rainbow

[2.] A. Melek and M. MacKinnon, "2006 Global Security Survey. Research Report" Deloitte2006. http://www.deloitte.com

[3.] H. Okhravi and S. Bak, "Colored LINUX: Covert Channel Resistant OS Information Flow Security",

[4.] **http://www.mobilec.org/**

[5.] **Trusted platform module** (tpm) summary **Https://www.trustedcomputinggroup.org/../trusted_platform_module_tp.**

[6.] S. Zander, G. Armitage, P. Branch, "Covert Channels and Countermeasures in Computer Network Protocols", IEEE Communications Magazine, vol. 45, issue 12, pp. 136-142, December 2007.

[7.] H. Qu, P. Su,and D. Feng, "A typical noisy covert channel in the IP protocol", In Proceedings of the 38th Annual International Carnahan Conference on Security Technology, pp. 189-192, 2004.

[8.] S. Li and A. Ephremides, "A covert channel in MAC protocols based on splitting algorithms", IEEE Wireless Communications and Networking Conference, vol. 2, pp. 1168-1173, 2005

[9.] C. Tsai, V. Gligor and C. Chandersekaran, "On the Identification of Covert Storage Channels in Secure Systems", IEEE Transactions on Software Engineering, vol. 16, no. 6, pp. 569-580, 1990

[10.] FIPA communicative Act Library Specification. "Foundation for Intelligent Physical Agents", 2000. http://www.fipa.org/specs/fipa00037

[11.] 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,