Efficient Mac Enforcement Approach For Intruders In Commercial Operating System

Mr.J.Satheesh kumar[#1] B.E.,(M.E), Mr.V.Dinesh,[*2]M.E.,
[#1] Student, Department of CSE, RatnaVel Subramaniam College of Engineering and Technology, Dindigul Dt,Tamilnadu,India.
[#1]inspiration.vdm@gmail.com
[*2] Assistant Professor, Department of CSE, RatnaVel Subramaniam College of Engineering and Technology, Dindigul Dt,Tamilnadu,India.
[*2]dinesh640@gmail.com

ABSTRACT- The firmest barriers to apply MAC to defeat malware programs are the incompatible and unusable problems in existing MAC systems. It's difficult to avoid malware problem in the commodity os enforce a practical access control approach to tackle the malware problems. Design a novel MAC enforcement approach, named Tracer, which incorporates intrusion detection and tracing in a commercial operating system. The approach conceptually consists of three actions: detecting, tracing, and restricting suspected intruders. The other is that, rather than restricting information flow as a traditional MAC does, it traces intruders and restricts only their critical malware behaviors, where intruders represent processes and executables that are potential agents of a remote attacker. Our prototyping and experiments on Windows show that Tracer can effectively defeat all malware samples tested via blocking malware behaviors while not causing a significant compatibility problem.

***IndexTerms- Acc*essControl, Precip, Malware, IntrusionDetection**

I.INTRODUCTION

Although protecting confidential information has long been recognized as one of the most important security problems,never before has the demand for its practical solutions been so imperative. A recent study by Webroot revealed that about 89% of computers it scanned were infected with spyware, with an average of 30 instances per machine This threat can be mitigated by the techniques which aim at preventing spyware from being installed or detecting and disinfecting spyware-riddled hosts However,reliance on these techniques as the only defense is risky, as evasion of them leaves confidential information completely unprotected. Therefore, it is crucial to enable a host to contain spyware surveillance, preventing data from being stolen even after attackers manage to breach other layers of defense. Unfortunately, this cannot be achieved by the access control mechanisms running in current commercial systems: for example, though mainstream word processing software such as Microsoft Word offers password and encryption protection to sensitive files, a keylogger can easily get around such defense by recording the password used by an authorized party to access these files. Fundamentally,these mechanisms are designed to regulate access to resources, not to control propagation of the information released from the resources Complementary to these mechanisms are the technologies for information flow security The idea is to track and manage sensitive data to prevent them from flowing into unauthorized parties. Research in this area started with the famous Bell-LaPadula (BLP) model The BLP model is designed to regulate the information flows between subjects (e.g., processes) and objects (e.g., files) with different sensitivity levels. Informally, the model forbids a subject from reading objects with higher sensitivity levels, or writing objects with lower sensitivity levels.However, these properties can be too restrictive for many commercial systems, in which most applications are multitasked and expected to work concurrently on the objects with various sensitivity levels. Moreover, BLP does not model the resources and input devices shared between sensitive subjects and public subjects (e.g., clipboard, screen and keyboard), which are widely used by spyware to gather sensitive information from the user. More recent research on information flow security focuses on tracking and controlling information flows within a program Many proposals require modifying source code to enhance it with information-flow policies. However, the source code of commodity software is usually not publicly available. we propose a novel MAC enforcement approach, Tracer, which consists of three actions: detection, tracing, and restriction. Each process or executable has two states, suspicious

or benign. An executable in this paper represents an executable file with a specific extension, such as .EXE, .COM, .DLL, .SYS,.VBS,.JS, .BAT, or a special type of data file that can contain executable codes, say a semi executable, such as .ZIP, .RAR,.DOC, .PPT, .XLS, and .DOT. The actions of detection and tracing change the state of a process or executable to suspicious if it is suspected to be malicious, and the entity marked as suspicious is called a suspicious intruder. The action of restriction forbids a suspicious intruder to perform malware behaviors in order to maintain confidentiality,integrity, and availability of the system, as well as to stop malware propagation. To be precise, once detecting a suspicious process or executable, Tracer labels it to be suspicious and traces its descendent and interacted processes,as well as the executables it generates. Tracer does not restrict any operations of benign processes. Meanwhile,it permits suspicious processes to run as long as possible but only forbids their malware behaviors.The novelty of Tracer is that, it incorporates light-weight intrusion detection and tracing techniques for configuring security labels, i.e., labeling suspicious OS entities, which is often done manually. Moreover, rather than restricting information flow as a traditional MAC does, it traces suspected intruders and restricts the malware behaviors of suspected intruders, i.e., processes and executables that are potential agents of remote attackers. These novelties lead to two advantages. First, Tracer is able to better identify potentially malicious OS entities and regulate their behaviors,which in turn significantly reduces the FP rate which is the root cause of incompatibility in existing MACenforced systems. Second, Tracer is able to label OS entities automatically to tackle the low usability problem which is the other major issue of existing MAC systems [2].We have implemented Tracer on Windows and have been using evolving prototypes of the Tracer system in our lab for a few months. Our experiments on the function of Tracer with a set of real-world malware samples demonstrate that it can effectively block malware behaviors while offering good compatibility to applications and good usability to normal users.Moreover, we have added another experiment to compare Tracer with existing practical online malware defense technology. The result shows that Tracer causes much fewer FPs than commercial antimalware tools and Mandatory Integrity Control (MIC) which is a MAC mechanism on Windows Vista.We investigate the root reasons of incompatibility and low usability problems of existing MACs. Although not all the observations are brand new, we believe that understanding these reasons more comprehensively and illustrating them through the design of an actual system are useful for other MAC researchers.

II. Related Work

Access

The ability to make use of information stored in computer system. Used frequently as a verb, to the horror of grammarians.

Access control list

A list of principals that are authorized to have access to some object.

Authenticate

To verify the identity of a person (or other agent external to the protection system) making a request.

Authorize

To grant a principal access to certain information.CapabilityIn a computer system, an unforgeable ticket, which when presented can be taken as incontestable proof that the presenter is authorized to have access to the object named in the ticket.

Certify

To check the accuracy, correctness, and completeness of a security or protection mechanism. Complete isolation.A protection system that separates principals into compartments between which no flow of information or control is possible.

Confinement

Allowing a borrowed program to have access to data,while ensuring that the program cannot release the information.

Descriptor

A protected value which is (or leads to) the physical address of some protected object.

Discretionary

(In contrast with nondiscretionary.) Controls on access to an object that may be changed by the creator of the object.

Domain

The set of objects that currently may be directly accessed by a principal.

Encipherment

The (usually) reversible scrambling of data according to a secret transformation key, so as to

make it safe for transmission or storage in a physically unprotected Environment.

Grant

To authorize (q. v.).

Hierarchical control

Referring to ability to change authorization, a scheme in which the record of each authorization is controlled by another authorization, resulting in a hierarchical tree of authorizations.

List-oriented

Used to describe a protection system in which each protected object has a list of authorized principals.

Password

A secret character string used to authenticate the claimed identity of an individual. Permission
A particular form of allowed access, e.g., permission to READ as contrasted with permission to WRITE.
Prescript A rule that must be followed before access to an object is permitted, thereby introducing an opportunity for human judgment about the need for access, so that abuse of the access is discouraged.

Principal

The entity in a computer system to which authorizations are granted; thus the unit of accountability in a computer system.

Privacy

The ability of an individual (or organization) to decide whether, when, and to whom personal (or organizational) information is released.

Propagation

When a principal, having been authorized access to some object, in turn authorizes access to another principal.

Protected object

A data structure whose existence is known, but whose internal organization is not accessible, except by invoking the protected subsystem (q.v.) that manages it.

Protected subsystem

A collection of procedures and data objects that is encapsulated in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem and the procedures may be called only at designated domain entry points.

Protection

1) Security (q.v.).

2) Used more narrowly to denote mechanisms and techniques that control the access of executing programs to stored information.

Protection group

A principal that may be used by several different individuals.

Revoke

To take away previously authorized access from some principal.

Security

With respect to information processing systems, used todenote mechanisms and techniques that control who may use or modify the computer or the information stored in it.

Self control

Referring to ability to change authorization, a scheme in which each authorization contains within it the specification of which principals may change it.

Ticket-oriented

Used to describe a protection system in which each principal maintains a list of unforgeable bit patterns,called tickets, one for each object the principal is authorized to have access.

User

Used imprecisely to refer to the individual who is accountable for some identifiable set of activities in computer system.

III Existing Methods

Mandatory Access Control (MAC) works without relying on malware signatures and blocks malware behaviors before they cause security damage. Even if an intruder manages to breach other layers of defense, MAC is able to act as the last shelter to prevent the entire host from being compromised. However, as widely accepted existing MAC mechanisms built in commercial operating systems (OS) often suffer from two problems which make general users reluctant to assume them. One problem is that a built-in MAC is incompatible with a lot of application software and thus interferes with their running and the other problem is low usability,

Which makes it difficult to configure MAC properly .Thus, enforcing a practical MAC on commercial OS to defend against malware is a promising but challenging task. All malware samples break into hosts through two entrances, network and

removable drive. Most breaking are via network through frequently used protocols such as HTTP and POP3. Malware behaviors can impose multiple forms of damages, i.e., resulting in problems in confidentiality, integrity, and availability. Besides, we consider malware propagation as another type of damage since it can indirectly cause the former three forms of damages and eventually lead the entire host to be taken over.

For example, the behavior "Copy itself" does not directly hurt security but is an essential step toward propagating itself and then executing malicious behaviors on a host. Therefore, we evaluate the damages of each behavior and record them, using C, I, A, and P to represent the damages related to confidentiality, integrity, availability, and propagation, respectively. Malware samples from the network have two attack patterns. One is that, most malware samples exploit bugs in network-facing daemon programs or client programs to compromise them, then immediately spawn a shell or back-door process. Next, an attacker typically tries to download and install attacking tools and rootkits, as well as performs some other adversary behaviors.The other attack pattern is that, malware samples increasingly use social engineering methods to lure users into downloading and launching them. After started, a malware sample usually copies itself and makes itself a resident in a host.

Limitations of existing system

- Less usability
- Original users are also blocked
- False positives increases
- Higher delay due to increase of false positves

IV Designing Method

Introduce Tracer, a novel MAC enforcement approach which integrates intrusion detection and tracing techniques to disable malware on a commercial OS in a compatible and usable manner. Through implemented Tracer on Windows OS to disable malware timely without need of malware signatures.

Developing a prototype on Windows is important, because most of the over 236,000 known malware items are designed for the attacks in the Windows environment, only about 700 malware items target for the attack of various Unix/Linux distributions.Based on the analysis of 2,600 malware samples, we extract 30 critical malware behaviors and summarize three useful malware characteristics, which will benefit future antimalware researches and investigate the root reasons of incompatibility and low usability problems of existing MACs. Although not all the observations are brand new, believe that understanding these reasons more comprehensively and illustrating them through the design of an actual system are useful for other MAC researchers.

Malware samples

Malware contribute to most Internet security problems. Antimalware companies typically receive thousands of new malware samples every day. An analyst generally attempts to understand the actions that each sample can perform, determines the type and severity of the threat that the sample constitutes, and then forms detection signatures and creates removal procedures. Symantec Threat Explorer is such a publicly available database which stores the analysis results of thousands of malware samples from various sources and is thus valuable to malware researchers. To have a thorough understanding of the philosophies behind malware design spent considerable amount of time analyzing the behaviors of malware programs. Specifically, since 2008, read, recorded, and analyzed the technical details of 2,600 malware samples of a wide range of formats and varieties, such as viruses, worms, backdoors, root kits, and Trojan horses.

As taking many samples from the same malware family might make the analysis results biased, have intentionally not chosen multiple samples of a polymorphic malware or similar malware. Malware behaviors can impose multiple forms of damages, i.e., resulting in problems in confidentiality, integrity, and availability. Besides, we consider malware propagation as another type of damage since it can indirectly cause the former three forms of damages and eventually lead the entire host to be taken over. For example, the behavior "Copy itself" does not directly hurt security but is an essential step toward propagating itself and then executing malicious behaviors on a host. Therefore, we evaluate the damages of each behavior and record them in confidentiality, integrity, availability, and propagation, respectively.

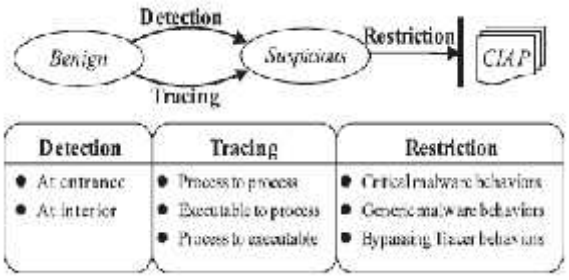
Tracer detection

The detecting action is responsible for identifying all potential intruders. We do not intend to design a complex intrusion detection algorithm to achieve a low FP rate at the cost of heavy overhead.

Instead, we design a light-weight intrusion detection algorithm that can identify all potential intruders but may have a relatively higher FP rate at the initial step. However, even if the detecting action wrongly The non dangerous protocols are difficult to be exploited by malware programs, because they are not permitted by firewalls since benign software rarely uses them.

Here, assume a "deny" default action for firewalls, thus any traffic not specifically allowed by firewall rules are denied. Nevertheless, in order to completely monitor all the network traffic, we denote a process as suspicious if it receives network traffic through a non dangerous protocol and then exhibits any of the malware behaviors. Instead of only checking non dangerous network protocols, further checking malware behaviors can reduce the extra high FP rate.

The Attack-Characteristics summarized in this point. That is, a process exploited by a malware program from the network necessarily executes at least one critical malware behavior, e.g., launching a shell process or downloading an executable, to propagate the malware program within the system. Although a carefully crafted malware program that subverts a process through a non dangerous protocol can perform some behaviors before performing a malware behavior, it is difficult for the process to make significant damages on the system.



The reason is that the malware behaviors monitored by Tracer include all of the behaviors that can cause significant damages, let alone that malware programs are difficult to attack a host through non dangerous protocols which are usually blocked by firewalls. The other type of entrances through which malware programs get into the system is removable drives according to the Entrance-Characteristics, hence we denote a process as suspicious when it opens or loads an executable from a removable drive.

Tracing intruders

To track intruders within an operating system, one can use OS-level information flow as done in However, a major challenge for leveraging OS level information flow to trace suspicious entities is that, file and process tagging usually leads the entire system to be A process receiving data from a suspicious process through a dangerous IPC;

A process reading a semi executable or script files with a suspicious label. For tagging processes, we observed that the excessive number of tags mainly come from tracing Inter process Communication i.e., marking a process as suspicious if it receives IPC data from a suspicious process, just as the approaches assumed in To address this issue, Tracer only tags a process receiving data from dangerous IPCs that can be exploited by a malware program to take control of the process to perform arbitrary malicious behaviors. Note that, dangerous IPCs do not include the other types of vulnerable IPCs that can be used to launch denial-of-service attack, or disclose sensitive information, escalate the privileges of the processes which send IPC data. Moreover, a dangerous IPC only involves the local IPCs instead of the IPCs over the network.

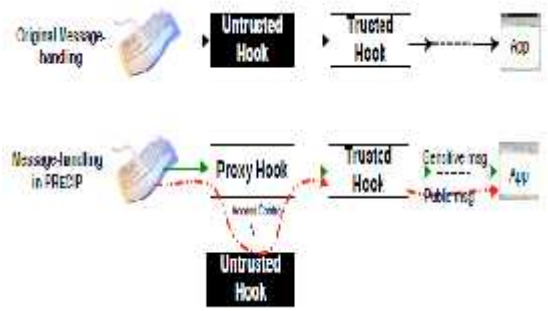
Figure 2. Tracer model

Since the detection at entrance can mark a process that receives IPC data from the network as suspicious. In order to identify the dangerous IPCs, we investigated Microsoft Security bulletins a database storing information about security vulnerabilities on Windows family OS and other Microsoft software. As malware programs usually exploit these vulnerabilities to compromise Windows hosts, Microsoft Security Bulletins become primary sources for analyzing attack vectors of Windows OS as done in Concretely, analyzed all vulnerabilities recorded in security bulletins related to named-pipes, local procedure calls, shared memories, mail slots,

and Windows messages from 1998 to 2009, as these IPCs send free-formed data that can be crafted to exploit bugs in the receiving process. However, among all of the security bulletins, only found one dangerous IPC, i.e., MS03-025 . The result reveals that in reality it is quite difficult to propagate malware through local IPCs within a Windows OS since people could only find one dangerous IPC over the period of 11 years. Consequently, Tracer employs a Dangerous-IPC-List to record and trace each type of dangerous IPC since there should be a very limited number of dangerous IPCs in a Windows OS.

Restricting intruders

In order to disable malware programs on a host, the restricting action monitors and blocks intruders' requests for executing critical malware behaviors listed in the principle of complete mediation for building a security protection system, Tracer further restricts two extensive behaviors, called generic malware behaviors, to protect security more widely.The first one is "Steal confidential information," which represents all illegal reading of confidential information from files and registry entries. The other is "Damage system integrity," which represents all illegal modifications of the files and registry entries that require preserving integrity. In addition, other behaviors that can be used to bypass Tracer mechanism also need to be monitored and restricted, including "Change file attributes," "Change registry entry attributes," "Execute non executable files," and "Execute Tracer special system calls."The behavior "Change file attributes" represents changing file extension names to executable or changing file DAC information. To efficiently restrict these malware behaviors, two issues need to be addressed. The first is how to determine the generic malware behaviors. Also identify behaviors "Steal confidential information" and "Damage system integrity" by monitoring illegal reading on read-protected objects and illegal writing on write-protected objects, respectively.



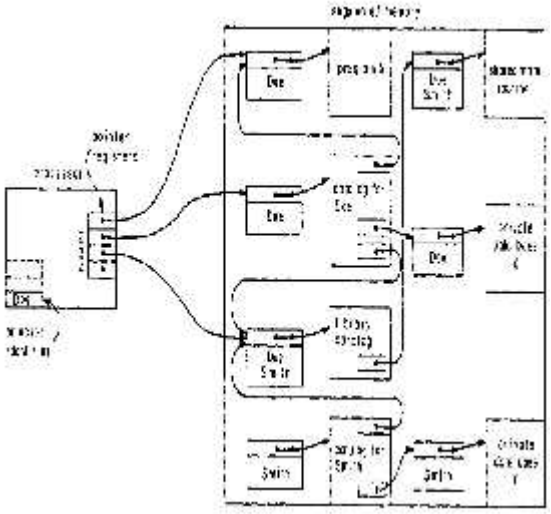
However, it is difficult to identify the objects that need protection among a large number of candidates in a Windows OS in order to recognize the generic malware behaviors. A traditional MAC requires users to give every object a security label to identify whether the object needs protection, which in turn becomes a heavy burden on general users. In Tracer, we use the DAC information of an object to determine whether it is protected. To be specific, a file, directory, or registry key is treated as read-protected when the user group "users" does not have a read permission on it. A file, directory, or key not readable by "users" means that it should not be readable by the world, and thus should be read-protected. Similarly, a file, directory, or key not writable by "users" is treated as write-protected. For other types of objects, e.g., IPC objects and system devices, we use "everyone" group to recognize protected objects.

Access controlled data usage

If the detection is purely based on known malware characteristics and behaviors, a detector may not be able to function effectively in the long run as new malware characteristics and behaviors may emerge over the time. To address this limitation a novel extensible mechanism is implemented in Tracer so it can dynamically add in new behaviors to monitor. A behavior consists of operation, object, and parameter.An operation is an abstract of one or several system calls with similar functions. For example, the operation create_ file corresponds to two system calls: NtOpenFile and NtCreateFile. In contrast, a single system call may contain more than one operation. For example, NtOpenFile contains four operations: read_file, write_file, create_file, and delete_ file.

The object and parameter of a behavior are extracted from a related system call. to dynamically add malware behaviors. In each concerned system call, we set up one or more checkpoints, each of which is responsible for checking the behaviors belonging to the same operation with the support of a modifiable behavior list in memory. The new malware behaviors are read from a configuration file and distributed to proper behavior lists corresponding to different operations in memory.

At each checkpoint, Tracer searches for the object and parameter currently requested in the corresponding list to determine whether the current access forms a malware behavior.



Advantages:

Tracer is able to better identify potentially malicious OS entities and regulate their behaviors, which in turn significantly reduces the FP rate which is the root cause of incompatibility in existing MAC enforced systems.

Tracer is able to label OS entities automatically to tackle the low usability problem which is the other major issue of existing MAC systems.

V Conclusion

A novel MAC enforcement approach that integrates intrusion detection and tracing to defend against malware in a commercial OS. We have extracted 30 critical malware behaviors and three common malware characteristics from the study of 2,600 real-world malware samples and analyzed the root reasons for the incompatibility and low usability problems in MAC, which will benefit other researchers in this area.

Based on these studies, we propose a novel MAC enforcement approach, called Tracer, to disable malware timely without need of malware signatures or other knowledge in advance. It detects and traces suspected intruders so as to restrict malware behaviors. The novelty of Tracer design is twofold. One is to use intrusion detection and tracing to automatically configure security labels. The other is to trace and restrict suspected intruders instead of information flows as done by traditional MAC schemes. Tracer doesn't restrict the suspected intruders right away but allows them to run as long as possible except blocking their critical malware behaviors.

This design produces a MAC system with good compatibility and usability. We have implemented Tracer in Windows OS and the evaluation results show that it can successfully defend against a set of real-world malware programs, including unknown malware programs, with much lower FP rate than that of commercial antimalware techniques.

REFERENCES

[1] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: Capturing System-Wide Information Flow for Malware Detection and Analysis," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 116-127, 2007.

[2] N. Li, Z. Mao, and H. Chen, "Usable Mandatory Integrity Protection for Operating Systems," Proc. IEEE Symp. Security and Privacy (SP '07), pp. 164-178, 2007.

[3] T. Fraser, "LOMAC: Low Water-Mark Integrity Protection for COTS Environments," Proc. IEEE Symp. Security and Privacy (SP '00), pp. 230-245, 2000.

[4].Microsoft,MandatoryIntegrityControlhttp://en.wikipedia.org/wiki/Mandatory_Integrity_Control, 2012.

[5] X. Wang, Z. Li, J.Y. Choi, and N. Li, "PRECIP: Towards Practical and Retrofittable Confidential Information Protection," Proc. 15th Network and Distributed System Security Symp., 2008.

[6].Symantec,Inc.,http://www.symantec.com/business/security_response/threatexplorer/threats.jsp, 2012.

[7] W. Sun, R. Sekar, G. Poothia, and T. Karandikar, "Practical Proactive Integrity Preservation: A Basis for Malware Defense," Proc. IEEE Symp. Security and Privacy (SP '08), 2008.

[8] K.J. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report MTR-3153, MITRE, Apr. 1977.

[9] L. Badger, D.F. Sterne, D.L. Sherman, K.M. Walker, and S.A. Haghighat, "Practical Domain and Type Enforcement for UNIX," Proc. IEEE Symp. Security and Privacy (S&P), pp. 66-77, 1995.

[10] P.-C. Cheng, P. Rohatgi, C. Keser, P.A. Karger, G.M. Wagner, and A.S. Reninger, "Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control," Proc. IEEE Symp. Security and Privacy, pp. 222-230, 2007.

[11] M. Howard, Fending Off Future Attacks by Reducing Attack Surface, http://msdn.microsoft.com/en-us/library/ms972812. aspx, 2003.

[12] M.Oers, OSX Malware Not Taking Off Yet,http://www. avertlabs.com/research/blog/index.php/2007/03/20/ osxmalware-not-taking-off-yet/, 2007.

[13] J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," Comm. ACM, vol. 17, no. 7, 1974.