

A Global Marking Scheme for Tracing Cyber Attacks

Yacine DJEMAIEL
CNAS Res. Lab (SUP'COM)
University of Carthage, Tunisia
yacine.djemaiel@laposte.net

Noureddine BOUDRIGA
CNAS Res. Lab (SUP'COM)
University of Carthage, Tunisia
nab@supcom.rnu.tn

ABSTRACT

Tracing complex attacks is among the research topics that are currently under development. Limiting tracing to network traffic has allowed the reconstruction of the attack paths of a few attacks, but appears to be insufficient to trace complex attacks. In this paper, we propose a new tracing scheme that extends marking to additional malicious activities related to system running processes and modification actions operated at the host level, making use of compromise independent disk based components. These components are involved in the marking and the tracing process. The behavior of the new scheme for marking and tracing is illustrated against a sample attack scenario that integrates several techniques in order to increase the complexity of the attack. Our scheme plays an important role in investigation and provides evidences that help an investigator determining the attacker and the actions he performed.

Keywords

Global marking, malicious activity, security attack, digital investigation

1. INTRODUCTION

Due to the increasing complexity of cyber attacks that cause damages to information systems and network components, investigation activity becomes a great need to determine the attack scenario and consequently the real attack source. To fulfill this need, the research community has focused studies on this field and many IP traceback techniques have been developed. The aim of these techniques is to determine the path to the attack source and base their processing on the network traffic. The techniques have shown their efficiency against a set of attacks such as denial of service attacks (DOS). However, these schemes are unable to determine the real attack source when there are mechanisms implemented at the attacker network such as the network address translation or when using spoofed traffic, which is the case for the most common distributed DOSs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07 March 11-15, 2007, Seoul, Korea.
Copyright 2007 ACM 1-59593-480-4/07/0003 ...\$5.00.

To overcome these limitations, some works have defined a marking scheme that is able to give more evidences and help identifying the attack source even when the intruder uses advanced techniques. The aim of this paper is to extend marking to additional intruder activity in order to render the investigation process more efficient and help deciding on the real attack source based on the set of marks associated with an extended malicious activity domain that includes network traffic, system processes handling, and modifications made to the *File Allocation Table* (FAT).

The rest of this paper is organized as follows. In Section 2, the related work in IP traceback field is introduced and the need to extend the marking process to enhance tracing and make them more efficient against complex attacks is described. Section 3 introduces the new marking scheme and discusses its components and mark structure. In Section 4, we present how to link the different kinds of marks and describe the tracing process according to the new technique. The following section presents the new scheme behavior performed against an attack scenario that includes many hacker tactics. Finally, Section 6 concludes the paper.

2. RELATED WORK

Recently, many IP traceback approaches have been proposed to determine an attack path. These techniques are classified according to the nature of processing performed on network traffic. Some proposed approaches (such as the hash based approach described in [7]) have made the choice to log information about the forwarded traffic. Additional solutions to the IP traceback problem addressed the storage of information that is called mark within IP packets. Approaches building marks are classified into two major categories: a) the probabilistic packet marking (PPM) approaches, which mark network traffic if a condition is fulfilled; and b) the deterministic packet marking (DPM) approaches, which mark all the network traffic before forwarding it to the next node.

DPM approaches observe some shortcomings related to the insufficient storage space within packets and the additional router overhead due to the marking of all packets. Moreover, PPM approaches present the same storage space limitation, in addition to the large number of packets that must be processed to reconstruct the attack path. In [6], several variations of PPM approaches have been developed to overcome these limitations. For deterministic approaches, an enhancement to the marking process is proposed by marking only at a specific location such as the network entry point, as described in [1]. This technique suf-

fers from the introduced router processing overhead and the inability to handle some complex attacks and some types of network traffic (e.g., encrypted packets). To contribute to this activity, we have proposed, in [3], a marking technique, called Adaptive and Selective Packet Marking (ASPM) that marks network traffic by applying an adaptive and selective scheme. Moreover, a DPM scheme based on redundant decomposition (DPM-RD) has been proposed in [4]. It introduces some modifications to the DPM described in [1]. These modifications concern the marking, recovery algorithm, and mark structure. Another alternative to enhance tracing is described in [5], where a distributed-log-based scheme, combining PPM and logging techniques, is proposed.

In addition to marking schemes, we have to give some details about a Cooperative Intrusion Detection and Tolerance System (CIDTS) that we have proposed in a previous work [2]. CIDTS detects and tolerates attacks by the cooperation of several components at the network, host and disk level. It contains a component, called the *system call interceptor* that ensures cooperation at the host level and introduces modifications to storage requests before forwarding them to the disk. These modifications consist in introducing information about the process (e.g. process identifier) associated to the storage request.

Most detection and tolerance capabilities are associated to compromise independent components that are located at the disk level. Among these components, we have added the *request handler* to handle incoming storage requests after checking them against a set of rules by another component referred to as the *violation rules monitor*. The rules and generated alerts are stored in a special area of the disk that is called the *protected area*.

3. GMS: THE MARKING PROCESS

In this section, we describe the new proposed marking scheme, referred to as *Global Marking Scheme* (GMS). We first describe the basic components used for marking and those responsible for mark management. Second, we discuss the mark structure, added to storage requests, and the mark protection.

3.1 GMS components

GMS interferes at different levels to enable global tracing. Consequently, the marking should be enabled by components that are located at the network, the system and the disk level. Figure 1 illustrates all the GMS components ensuring marking and tracing. The network traffic marking is performed at the network entry point router's while the processes activity is marked at the system level by the *system call interceptor* that has been defined in [2]. The extraction and storage of marks is performed by the *request handler* that is a compromise independent component located at the disk side as described in [2]. *TraceSrc* is a module that performs the mark extraction, storage and validation for network traffic. This component is introduced into an intrusion detection system. As an enhancement to the ASPM marking scheme, this module can be integrated at the disk level to introduce additional protection to the marking process, since it would be performed by a compromise independent component. Actions performed by the *TraceSrc* module will be handled by the *request handler*.

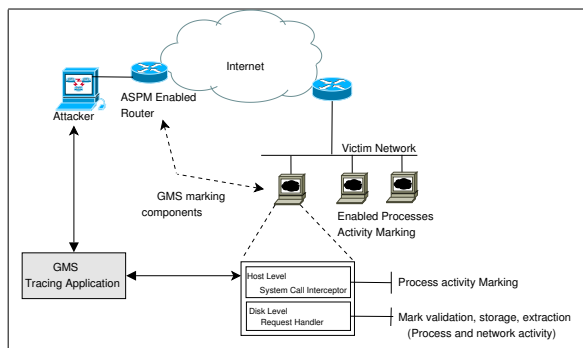


Figure 1: GMS components

3.2 Mark structure

GMS uses two kinds of marks: marks associated to network traffic and marks assigned to the process activity and the modifications introduced to the FAT. The structure of the first type of marks contains a series of information including the IP address of the marking interface as well as the marking date and time. A HMAC digest is appended to the mark and calculated based on the content of non mutable fields of the traffic unit. For an IP traffic, the content of the following non mutable fields is used: the protocol type, sequence number, TCP source port, and the destination port.

The second kind of marks includes information that identifies the user and the process into the storage requests. These information may be represented respectively by the user identifier, the process, and its parent identifiers. We have found a mean to associate storage requests to users and processes by the enhancement of the commands that will be forwarded to the disk. These modifications concern the introduction of the identifier of the process and its parent in addition to the decision made by the *system call interceptor*. A mark added to storage requests encapsulates the following information: the user ID, the process ID and the parent process ID.

3.3 GMS data structures

To perform marking, we need to define a set of structures to hold received marks for post-mortem operations, such as tracing the attack source for investigation needs. For network traffic marking, we have introduced a modification that is described in Section 3.1 by adding the *TraceSrc* module to the disk side. Thus, it is more interesting to move the extracted marks stored into the *trace database* (*TraceDB*) to the disk *protected area*. The marks extracted from network traffic and the content of the non mutable fields associated to them will be stored in a file called *network marks file*.

Another structure is needed to handle inter process communication that is called *process dependency file*. This structure is stored in the *protected area* and joins processes having exchanged data before performing actions on other resources (e.g., files, directories). An entry associated to the *process dependency file* holds information about the identifiers associated to communicating processes, the request time and associated marks involved in this communication process.

Marks associated to processes activity are handled by the *request handler* and are stored in a secure manner in the *forensic log file* that is located at the *protected area*.

3.4 Mark protection

Protecting generated marks in GMS should fulfill the following set of constraints: a) a stored mark cannot be accessible by an intruder; b) the marking process associated to process activity should be performed by a kernel level component; c) a compromise independent component should control the marking module processing; and d) mark extraction and manipulation should be performed by compromise independent components.

Needing to collect information about users (i.e. performing marking at host level) and knowing that an intruder may be able to modify the marking process when the host is compromised, we have integrated into the *system call interceptor* code's a routine that monitors some variables related to this module and sends periodically a protected hash of this information to the disk side components. The *request handler* uses these digests to detect the compromise of the *system call interceptor*. The GMS components perform marking mainly at three levels: the network, the system and also the disk level.

Network activity marking:

The marking of network activity has been detailed in a previous work when describing the ASPM technique in [3]). The ASPM marking process is enabled at the close interface to the source of the packet on the edge ingress router. The basic idea of ASPM is to select and mark attack traffic that matches specific properties in order to reduce or avoid major IP traceback problems (e.g., processing overload and bandwidth overload). Marks are appended to network traffic by the ASPM enabled router before forwarding it to the next node.

Process activity marking:

Marking is performed according to two ways that depend on the process behavior. The first way is related to process activity in terms of read/write storage requests. For this, the *system call interceptor* inserts the mark, with the structure described in section 3.2, into the storage requests. The second way handles the inter process communication in order to monitor dependency between processes. To handle inter process communication, the *request handler* adds an entry to the *process dependency file* having a structure similar to what is described in section 3.3. Of course, this entry will hold the two *id_marks* associated to the actions done by the two communicating processes. The search operation of these two marks is based on the *request_time* of storage request generated for the second process. The *request handler* will determine the last request performed by the first process based on this temporal information (*request_time*).

In order to illustrate how the marking is performed for processes activity, we will consider the case where the first process called Pr_1 read $File_1$ content's and communicates the content to Pr_2 . Then, Pr_2 proceeds to the creation of $File_2$.

When the *system call interceptor* detects inter process communication, it sends a storage request to the disk indicating that the two processes are initiating a communication. In this request, the *system call interceptor* inserts the identity of the two processes communicating followed by a special block number indicating that there is a possible communication between the two processes. Reception of this special storage request activates the addition of a new entry to the *process dependency file* where only the mark id of the last request associated to Pr_1 will be inserted in this

entry waiting for the storage request related to Pr_2 .

When receiving the storage request associated to Pr_2 and before adding an entry to the *forensic log file*, the *request handler* checks whether there is an entry in the *dependency process file* that matches the process ID. If that is the case, a new entry is added to the *forensic log file* pointing to operations requested by Pr_2 . This entry will hold the content of the storage request, the mark content, and the time of request reception (*request_time*). After that, the *request handler* updates the *dependency process file* entries that match the communicating processes by adding the id of the mark associated to Pr_2 . Search of the mark can be performed using the *request_time* and the *process identifier*. The recent storage request associated to Pr_1 is determined as follows: $\min(\text{request_time}_{Process2} - \{\text{request_time}_{Process1}\})$.

Disk based marking:

For the foregoing marking types, both network and host based components are involved in the marking process. The CIDTS monitors another kind of operations performed at the host level that is: modifications introduced into the FAT. These modifications are forwarded to the disk side components in order to be stored in the *protected area* and that serve to trace the set of modifications introduced to the FAT. In order to enhance the tracing process, these modifications are marked using the same structure as for storage requests. As a result, each modification to the FAT is also associated to the user and the process (and its parent). Following the same principle as for storage requests, the occurrence time of each modification will be stored in addition to the mark content. Information on the user and the process associated to the FAT modifications are collected from the *forensic log file* based on information related to the set of storage requests that have the most near occurrence time and that reflect the content of the storage request. The user and process identifier are determined from the *forensic log file* based on a set of criteria such as the actions associated to the FAT modifications and the occurrence time.

Among all constraints that can be considered when marking and storing marks, there is a constraint related to storage space limitations. For the ASPM scheme, we adopt a selective marking scheme since resources are limited at the marking router. For both marking processes activity and FAT modifications, the marking module does not suffer from resource limitations that's why we mark all requests. At the other side, the storage of these marks is performed intelligently by the *request handler* in the way that the mark is added to the *forensic log file* for the first time associated to an identifier (*mark_id*) but if the next requests have the same mark, we insert only the *mark_id* in each new entry.

4. GMS: THE TRACING PROCESS

To perform the tracing according to the principles described in Section 3, we need to make some assumptions associated to the set of GMS components that should be deployed in order to be able to trace complex and advanced attacks. The set of major assumptions that should be satisfied are the following: 1) an ASPM enabled marking router should be available for each network entry point; 2) the ASPM marking router should not be compromised; 3) the CIDTS should be deployed at least at victim networks; 4) the victim networks should have deployed GMS components for process activity marking and mark management (disk side components); and 5) the clocks should be synchronized

at least for each site.

4.1 Linking the two-level marks

The tracing of attacks, which are performed locally by an intruder who gains a physical access to the compromised host, will only use marks that are associated to process activity. For the remaining cases, the use of both types of marks will be needed in order to trace complex attacks. If one analyzes the behavior of most common attacks, he would notice that a victim receiving malicious network traffic initiates processing that leads to system compromise or moving the system to a failure state. From this observation, it is possible to make a relation between the network activity and the set of actions performed by running processes. Therefore, one can say that the malicious network traffic is propagated to the system through running processes, which can be consequently considered malicious.

To link the two types of marks, we need to extract the information related to the requested service (e.g., destination TCP port) from marks associated to network traffic. Based on this information, the socket is determined since it is associated, by definition, to a port and a running process. At this level, it is possible to determine marks associated to processes involved based on the available information about the process and its parent. Linking processes activity marks to marks assigned to FAT modifications, which are added at the disk level by the *request handler*, is done in the same way and is based on the couple process and parent-process identifiers. At the end, it is useful to notice that the linking process may be performed in both directions, from network marks to marks associated to processes activity or reversely.

4.2 The tracing process

The tracing process is not performed automatically but initiated using a request generated by an authenticated user (e.g. system security administrator). Moreover, the tracing is performed based on the selection of an attack from a list of attacks extracted from the *alert file*. An external application, called *GMS tracing application*, can be used for tracing detected attacks. This application bases its processing on the collected data that is stored in the *protected area*. This application can access the *protected area* only in a read mode and this is will be possible after authenticating it and the user using it (e.g. using Kerberos, digital certificates, etc.). We have opted for an external application, in order to allocate external resources to the processing of these requests that generally require intensive resources to handle them and to make the necessary correlation between all marks collected at the victim host.

At the initial running of *GMS tracing application*, GMS accesses the *alert file* (containing the attack attempts generated at the network, host and disk level) generated by CIDTS. Based on its content, it displays a summary of attack attempts ordered by their occurrence time. Using this information, the administrator may select the attack that is asked to trace. The processing of the *GMS tracing application* depends on the kind of the performed attack. The two identified attack classes are: a) Attacks performed by gaining direct access to the system; and b) Attacks that are initiated remotely by an intruder.

Tracing attacks in the aforementioned classes uses the same set of performed actions that correspond to the tracing of processes activity and FAT modifications at the system

interacting with the *GMS tracing application*. Therefore, we only describe, in the following, the tracing for the second class of attacks that includes malicious activity at all levels (i.e., network, host and disk). In this case, the *GMS tracing application* checks marks held in the *forensic log file* and the *dependency process file* using, as searching criteria, the process identifier and the attack occurrence time retrieved both from the *alert file*. Marks that match the selected criteria will be used to construct a list of user identifiers, their related processes identifiers and the occurrence times. Having all participating processes in the requested attack, the next step will be to detect marks held in the *network marks file* that are associated to the identified processes activity.

The identification of marks can be made according to the linking principle described in section 4.1. At the end, the application displays the set of users and processes involved in the selected attack and also the IP addresses associated to the attacker. Details for the need of investigation may be provided at this level; but, this is not sufficient since the application should extract additional information from the *forensic log file* and the *process dependency file* to obtain an output that associated identified users and processes to malicious activity that may be represented by resources accessed or created during the attack process.

5. CASE STUDY

In this section, we describe an attack scenario that combines many hacker tactics to compromise the target and hide the real attack source. Based on the set of collected marks by GMS, we will illustrate the behavior of the proposed technique in order to trace the attack.

5.1 Attack scenario

Figure 2 illustrates the chosen attack scenario by which an intruder uses a compromised host to attack the victim network. Access to the compromised host is obtained by exploiting an available vulnerability by running a buffer overflow attack over the vulnerable FTP server. The attacker uses the compromised host as a zombie to perform an advanced attack against the victim. The attacker tries to collect information about accounts used to log to different hosts and servers located at the victim network. To get these access information, the attacker performs an SQL Injection attack by interacting with the authentication server through a set of forged SQL queries. The output associated to a specific selection query enables the intruder to gain access to victim network through available network services such as SSH (Secure SHell).

When the attacker logs to the victim host as a privileged user, he tries to hide his traces by downloading and installing a forged logging service and replacing used binaries (e.g. *ls*, *ps* for UNIX systems) by rootkits. After performing these malicious actions, he tries to collect some critical information about the victim such as service configuration files, password file, network parameters, etc. Then, the attacker installs backdoors to enable future visits to the system without launching again the same attack.

5.2 GMS behavior against a complex attack scenario

The tracing process is performed using the external *GMS tracing application* and the collected marks at all levels (network, system and disk). The application extracts first the

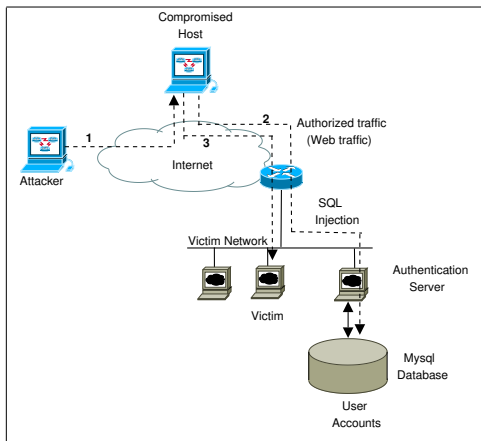


Figure 2: Attack Scenario

information about the attack and then displays them to the authenticated user using alerts generated by CIDTS. For this attack, marks are stored at two different locations: the compromised host used as a zombie and the victim site. At the compromised host, there are marks generated by the ASPM enabled router at the entry point to the network. These marks contain the IP address of the attacker. Moreover, the GMS components mark storage requests associated to the buffer overflow attack to identify the users and associated processes.

FAT modifications are marked by the *request handler*. In order to reference these marks (for both processes activity and FAT modifications), we put them in a set denoted by set#1. The next attack phase is more important for tracing using marks generated by the ASPM enabled router, which identifies the IP address of the compromised host. The available firewall (if it is not an application level firewall) does not block the SQL Injection attack traffic since it uses authorized traffic and the attacker is not visible. The set of marks associated to the process and user handling the forged SQL queries, which are stored at the authentication server, is denoted by set#2. The first alert that will be generated at the victim system will be associated to the download of malicious tools from the hacker remote site; but before that, activity associated to the login process are marked and associated to the adequate user and process. These marks are identified as set#3.

When starting malicious activity to hide traces and violate the victim local security policy, GMS components marks this activity and stores the associated marks in the *protected area*. These marks are put together to form set#4 and contain information about processes and users involved in this attack and marks associated to FAT modifications. Based on set#4 and the generated alerts, ordered in time and stored in a *protected area*, the external *GMS tracing application* interacts with these generated data to trace the attack as follows. First, it starts by finding the link between set#4 and set#3 marks since the latter is associated to a malicious activity. Then, it builds a link between the marks in set#3 and set#2 to determine at this level a new information about the attacker that is limited only to the IP address of the compromised host. The output of the external *GMS tracing application* will be limited to information that de-

termines the compromised host.

Having the IP address of the compromised host, it would be possible to run GMS against the compromised host to exploit the marks collected. In a similar manner, it is possible to determine the attack source (using ASPM marks) and the set of actions performed at the compromised host and associated to processes and users involved in this attack. Based on the two treatments performed at the victim and the compromised host, *GMS tracing application* is able to correlate the two results and constructs a unique output that links all these findings and gives a complete tracing result. This processing will be in a recursive way if the attack involves more actors.

6. CONCLUSION

In this paper, we have presented a novel marking approach that manages malicious activity to provide global tracing. The proposed marking scheme does not limit marking to the network traffic but extends it to process activity performed at the host and requests at the disk side. The marking technique marks network traffic by applying an adaptive and selective strategy. Moreover, it marks processes activity at the host level and FAT modifications at the disk side. Marks management is performed at the disk level exploiting an important feature that is: compromise independence. GMS is shown to be effective in digital forensic investigation. Adopting GMS will enhance both tracing and investigation capabilities and helps performing better correlation for the need of intrusion detection and attack identification.

7. REFERENCES

- [1] BELENKY, A., AND ANSARI, N. IP Traceback with Deterministic Packet Marking. In *IEEE COMMUNICATIONS LETTERS* (2003), vol. 7.
- [2] DJEMAIEL, Y., REKHIS, S., AND BOUDRIGA, N. Cooperative Intrusion Detection and Tolerance System. In *Proceeding of 12th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2005)*. (Tunis, Tunisia, December 2005), pp. 279–283.
- [3] DJEMAIEL, Y., REKHIS, S., AND BOUDRIGA, N. Marking and investigating communication traffic: an adaptive and selective scheme. In *WSEAS TRANSACTIONS on COMMUNICATIONS journal* (Athens, Greece, July 2005), vol. 4, pp. 469–477.
- [4] JIN, G., AND YANG, J. Deterministic packet marking based on redundant decomposition for ip traceback. In *IEEE COMMUNICATIONS LETTERS* (March 2006), vol. 10, pp. 204–206.
- [5] JING, Y., TU, P., WANG, X., AND ZHANG, G. Distributed-log-based scheme for ip traceback. In *The Fifth International Conference on Computer and Information Technology (CIT05)* (2005), IEEE Computer Society, pp. 711–715.
- [6] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Network support for ip traceback. In *IEEE/ACM Transactions on Networking* (2001), vol. 9, pp. 226–237.
- [7] SNOEREN, A., PARTRIDGE, C., SANCHEZ, L., JONES, C., TCHAKOUTIO, F., KENT, S., AND STRAYER, S. Hash-based ip traceback. In *Proceedings of ACM SIGCOMM 2001* (August 2001).