PRE-FORENSIC SETUP AUTOMATION FOR WINDOWS 2000

Flávio de Souza Oliveira
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
flavio.oliveira@ic.unicamp.br

Célio Cardoso Guimarães
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
celio@ic.unicamp.br

Paulo Lício de Geus
Institute of Computing
University of Campinas
Brazil
13083-970 Campinas - SP
paulo@ic.unicamp.br

ABSTRACT
This work presents a framework for automation of administrative tasks and deployment of protection mechanisms to facilitate a future forensic analysis. The main goal is to disclose and supply measures for a fast configuration of Microsoft Windows 2000 networks, when deploying incident response procedures.

KEY WORDS
Security, Forensics, Incident Response, File Integrity, Windows 2000, Automation

1- Introduction

Establishing procedures for incident response by companies interconnected via Internet is considered of vital importance in order to minimize financial losses when attacks succeed. Few institutions, however, have adopted such procedures, due to the high cost involved and technical difficulties for deployment. [1]

Preparing for a consistent response starts well before the incident, with the design of procedures and policies, responsibility assignment and personnel training. Procedures to be adopted during an eventual incident must be previously tested and rehearsed due to the great pressure involved when the incident occurs, which may lead to procedure errors that may render evidences useless and may impair the progress of the investigation.

An important stage while designing a response procedure is the correct configuration of the networked computers for an eventual forensic analysis. This is particularly important on Windows platforms, where default configurations leave very few footprints of user activities.

When a Windows machine is previously configured as a target for forensic analysis, the search for evidence is facilitated by correct utilization of resources that allow the collecting of facts occurred in the recent past of the attacked machine [2]. It can be extremely laborious, however, to configure large networks, especially Windows 2000 networks, which has few mechanisms for administrative task automation.

We have developed the tool PFSAF (*Pre-Forensic Setup Automation Framework*) with the goal of automating some administrative tasks that may significantly speed up a possible forensic analysis, and to provide mechanisms for magnifying footprints of illegal use of resources in Windows 2000 machines. PFSAF is a GPL software aiming at remote setup of W2k machines for future analysis.

This work aims at presenting relevant aspects of W2k machine configuration towards a future forensic analysis, and also at presenting a new framework for automation of tasks related to this configuration.

1.1- Related Work

While there are several works in the area of administration of large[1] Windows networks, very few focus on the needs of computational forensics. Furthermore, most tools for automation of administrative tasks in Windows NT/2000 networks are proprietary commercial tools.

Despite the scarcity of tools satisfying the needs described in this work, two works present similarities with our framework: the set of scripts presented by Harlan Carvey in [3], with similar features in the nucleus of PFSAF, such as remote query of shortcuts in the Startup folder. Those scripts, however, are not scalable to large networks: the administrator would have to edit and execute each script on all machines. The second work is DoIt4Me [4], which is a tool for automation of administrative tools in large Windows NT networks, but which does not cover all requirements presented in this work, such as file integrity.

Tripwire, a tool for file integrity verification, is also a related work. [5]

1.2- Incident Response

Security awareness is nowadays an essential requisite for the majority of network applications. However, the main problem to face is: even when all security measures are taken, security failures may occur, because some non dis-

1. Networks with several hundred or thousands of machines.

closed or unknown vulnerability may be exploited by a new attack. One can not assert, therefore, that no matter what security apparatus we have (Firewalls, VPNs, etc.), our system is immune to attacks. This is due to the fact that such apparatus as services provided through the Internet are composed of many software components with thousands of lines of code not immune to programming errors.

Assuming that there is no security scheme immune to failures, it is necessary to define procedures to be followed in case of a well succeeded attack, besides the availability of people to perform these functions (Response Team). Awareness with such methodologies is still rare within corporations.

2- Computer Forensics

The advent of the first digital crimes involving the computer environment, made necessary the creation of a new forensic discipline to act in this new niche, with methodologies and cumulative knowledge in the acquisition, handling and analysis of digital evidence.

The solution of a computational mystery can be an arduous and difficult task. The system must be meticulously examined, in the same way as a detective examines a crime scene [2]. For this purpose the person making the analysis must thoroughly know the operating system in order to identify and understand the cause-effect relationships of all actions taken during the analysis.

Cause-effect relationships are not sufficient, however; there is the need of more skills in order that a professional can effectively conduct a forensic analysis. Fortunately, according to Venema & Farmer [2], many of these skills are characteristic of programmers, such as logical reasoning and an open mind. Such skills are largely used during the search for the cause of errors in a program. Debugging a faulty program, however, is far away from the challenge of forensic analysis, because when someone debugs a program he is fighting against himself, while in forensic analysis one is challenging another programmer who is not interested in being discovered [2].

2.1- Digital Evidence Manipulation Standards

In this section we present an overview of the present stage of the computational forensic standardization efforts. The importance of such efforts is due to the need to guarantee the integrity of evidence presented to a court, since once standardized such procedures, it becomes legally impractical to deny the facts presented assuming that the methodology was correctly used while dealing with the proofs.

Despite the existence of several works in this area, we still note a scarcity of methodologies for the handling of evidence, at least as compared to other forensic disciplines [6].

There are some international standards being experimentally used [7]. They were developed by the SWGDE (*Scientific Working Group on Digital Evidence*), which is the American representative in the IOCE (*International Organization on Computer Evidence*). These standards were presented during the International Hi-Tech Crime and Forensic Conference, in London, 4–7th October 1999.

The standards developed by the SWGDE follow a unique principle: all organizations dealing with forensic investigation should keep a high quality level in order to assure the reliability and precision of evidence. This high quality level can be achieved through the elaboration of SOPs (*Standard Operating Procedures*), whith procedures for all types of known analysis and techniques, equipment and materials accepted by the international scientific community [7].

In Brazil, as an example, there is no standardization in progress, however, ongoing research in this area, can be seen in [6] and [8]. Also, there are some works done upon request of the Federal Police, aimed at non computer professionals such as prosecutors and federal judges.

3- W2k Forensic Analysis

It is important to configure machines in a way that takes into account a possible future forensics analysis. To fully understand the implications of the latter statement, one needs to know how an investigation on a W2k machine should be done. In this section we cite only a few examples of procedures that might be adopted, since each case presents its own peculiar needs, which accounts for the non-viability of defining procedures applicable to all possible situations that can be found during a security incident.

3.1- Live Analysis on W2k Machines

A live analysis can be defined as one performed on a system victim of a security incident, before any procedure is taken for shutting it down. This kind of analysis is extremely important for an investigation, given that it is the only opportunity to collect a series of volatile information, which otherwise would not be available after a machine restart. Among these are currently active network connections and programs currently running before initiating the investigation.

The biggest problem with this kind of analysis is the lack of control over the machine under analysis, since it can still be under control of an attacker. That means that a whole assortment of programs and libraries may be active, all developed to conceal information and to lure the investigator. For this reason, it is mandatory for this analysis to be performed from programs and libraries run from trusted media (e.g. CD-ROM), so that the examiner may have assurance of the integrity of the binaries being run.

Another possible problem the investigator is going to face, during the selection of tools that will make up his application CD (Response Kit), is how he is going to find out all library (DLL) dependencies, needed for those programs to run. One way to do that is through the use of *Dependency Walker (depends.exe).* This tool comes with W2k's Resource Kit and analyses all the dependency tree

of a given program, giving a break down of functions used by each library. An alternative is *listdlls.exe*[2], developed by Mark Russinovich. This tool is capable of listing all libraries that are currently being used by a process. As such, it is necessary first to execute the tool being analyzed and then to use listdlls.exe to obtain the DLL list. It can also be useful during a live analysis, where it can be used to analyze a suspect process on a broken-in machine.

This strategy of including all libraries needed to run Response Kit programs during a live analysis can minimize the use of code originated from the victim machine. However, this is not enough to guarantee that no DLL from the victim machine will be used. The reason is that in a live analysis, the machine is in the state when approached for the analysis, and as such a number of libraries are already loaded in memory. Any program that needs an already loaded library will not have the operating system load it again (known-to-be-good binary from CD), but rather use the currently loaded, and possibly compromised, one. This opens a window of opportunity for the production of false results.[9]

The most appropriate solution to avoid access to insecure dynamically-loaded libraries is to eliminate all dynamic access through static compilation of all required tools for the analysis. Nevertheless, due to the commercial focus of the Windows family, very few tools for this system are open source, which pretty much rules out this approach.

Another possible solution would be the removal of all DLLs present in memory prior to the analysis, but the collateral effects are hard to predict, since W2k does not present mechanisms to safely perform this. A possible outcome would be a number of GPFs (*General Protection Fault*) on currently executing processes, which could jeopardize the forensic analysis. One must recall that a fundamental principle in such procedure [2] is not to disturb the system under analysis.

A Windows 2000 live analysis represents a big problem, due to these problems and to others to be discussed in Section 3.3.

3.2- W2k Analysis Basis

Most of the main steps toward a computer forensic analysis can be ported to several operating systems. W2k is no different, with several currently adopted procedures being originated from other platforms, notably Unix.

After a careful live analysis is done, it may be necessary to shutdown the victim machine for a *postmortem* analysis, which may be described as an analysis performed under a controlled environment. In this, appropriate procedures should be done to act over the system and collect information from it at a low level, such as:

- File Slack: Microsoft's operating systems store their files on disk using fixed-size data blocks called *clusters*, however file contents have no restriction in size. Usually, the last *cluster* associated to a file is not fully utilized, which allows for data excluded from this and past files to be later captured and analyzed.
- RAM Slack: Besides data from previous files resident on disk, the *file slack space* can also hold sets of bytes randomly selected from RAM memory. This happens because Windows (as most other operating systems do) normally writes on disk in 512-byte blocks, called sectors. Since normally the amount of information to be written is not a multiple of 512, usually the last block of information will have to be padded to match a sector size. Windows uses its own memory buffers to get irrelevant bytes to do that.
- Alternate Streams: it can be said that every NTFS file holds another embedded file with no explicit name, called default stream or unnamed stream, where conventional data like text and programs are stored. Nevertheless, one can also create embedded files with different names, called alternate streams. The problem is that the detection of these embedded, but named files, cannot be natively effected by any W2k programs, enabling them to be easily used to conceal programs and information. [10]

The analysis must be conducted over image copies of the original disks from the victim machine, so that there is no risk of a mistake on the part of the examiner to compromise the integrity of the original files, causing irreparable damage to the ongoing investigation. This controlled environment could be, for instance, a machine with several operating systems containing adequate tools, such as the Foundstone Forensic ToolKit[3].

By using another machine with a Unix-like operating system that has support for the NTFS file system, such as Linux, one can take advantage of powerful tools developed for the Unix platform, such as TCT[4] (*The Coroners ToolKit*). Also, the media can be mounted read-only, whereas W2k would normally alter the index files of the partition during boot. Another advantage of using a platform like Linux for the *postmortem* analysis is the possibility of visualizing the evidence from another point of view.

Unfortunately, Linux's support for the NTFS filesystem is not complete, so that several data structures, such as the alternate streams, are ignored. This force at least part of the procedures to be performed under W2k itself.

As a general rule, one can say that the majority of conclusions from a forensic analysis is based on results obtained from a live analysis, seconded by an analysis of events registered by EventViewer and trailed by the file integrity checking. Not always a detailed postmortem over

2. http://www.sysinternals.com/ntw2k/freeware/listdlls.shtml

3. http://www.foundstone.com/knowledge/proddesc/forensic-toolkit.html
4. http://www.porcupine.org/forensics/tct.html

the filesystem is performed, or even needed. Often, it is not viable to create disk images due to size constraints on storage available in the analysis machine, or even due to the impact the analysis might impose on running services, such as stopping them altogether, which may not be acceptable [11].

3.3- W2k Forensic Analysis Problems

The forensic analysis of a proprietary system such as Windows 2000 makes it harder to define fully reliable methodologies, once it is not known for sure the exact consequences of all actions taken during its analysis [10].

Since W2k is an undisclosed-code system, it is not possible to prove with full certainty that no result contamination may have happened.

Another problem worth mentioning is that in the Windows environment the GUI culture is praised, which tends to hide location and treatment given to information being handled. This rarely happens on open source systems like Linux and FreeBSD when documentation is not available. The lack of such information makes it difficult gathering data from the target system without the use of tools originally available on the operating system. [2]

4- Pre-Forensic Setup

Despite current security concerns being essential requirement for a variety of network services and applications, it is still very common to find W2k networks with out-of-the-box configurations. These usually deal only with functional aspects, while security configurations are given a lower priority.

Take, for instance, the auditing policy configurations, which are not enabled by default, leaving all security-related events not to be logged.

The correct configuration of a machine, where a possible, future forensic analysis is taken into account, can substantially speed up the response to a security incident, and also possibly bring its solution closer. This is due to the fact that a correct configuration will force W2k to supply many more traces of an attacker's actions, further enlarging the collection of evidences that could draw the case to an end.

Below are discussed some examples of topics whose configuration or deployment could be of great benefit for the progress of an analysis:

- Audit Policies: Event auditing is an important part of network administration. When an administrator selects events to be audited, through the analysis of log files, he can later, derive normal system utilization patterns, possible security problems and observe trends in network resource utilization. Care, should be taken when choosing what is to be audited, given the large number of events some activities can generate and the overhead associated with them [12][8]. Table 1 shows a description of all auditable events on W2k.

Event Category	Description
Account logon events	Activated when a domain controller receives a logon request
Account management	Activated when a user account or group is created or changed
Directory service access	Activated when an Active Directory object is accessed
Logon events	Activated when a user logs on or logs off
Object access	Activated when an object is accessed
Policy change	Activated when a policy affecting security, user rights, or auditing is modified
Privilege use	Activated when a user right is used to perform an action
Process tracking	Activated when an application executes an action that is being tracked
System events	Activated when a computer is rebooted or shut down or another event occurs that affects security

Table 1: Auditing categories

- ACLs: NTFS stores an access control list (ACL) with every file and folder on an NTFS volume. The ACL contains a list of all user accounts and groups that have been granted access for the file or folder, as well as the type of access they have been granted. When a user attempts to gain access to a resource, the ACL must contain an entry, called an access control entry (ACE), for the user account or a group to which the user belongs. The entry must allow the type of access that is requested (for example, Read access) for the user to gain access. If no ACE exists in the ACL, the user cannot gain access to the resource. The ACL setup with a correct audit policy configuration, can show general tracks of system utilization.[13]
- File Integrity Check: It is very difficult to compromise a system without altering a system file, so file integrity checkers are an important capability in the search for evidences and tracks. A file integrity checker computes a checksum for every critical file and stores this. At a later time you can compute a checksum again and test the current value against the stored value to determine if the file has been modified.[12][5]

5- Pre-Forensic Setup Automation Framework

Contrary to Unix systems, Windows is considered a "hands-on" system, which requires the administrator's presence or actuation on each machine, so that most installations, configurations and administrative tasks are performed on a network. This makes the administration of a large Windows network an extremely laborious and unproductive task.

The administrative hardships imposed by the Windows platform are a severe obstacle for the deployment of the

recommendations on Section 4. They, in fact, cause such deployment to take quite a long time from the Response Team. In this section we present a framework implementation (PFSAF) that seeks to automate the main actions required to prepare machines on a network for the chance of a future forensic analysis.

5.1- Structure

PFSAF was structured to allow all necessary configurations from a single machine, and also being able to schedule tasks through Task Scheduler at a more convenient time for the administrator. This machine (Forensic Station) must be of restricted access and will be responsible for the storage of all configuration files and databases generated by PFSAF (Figure 1).

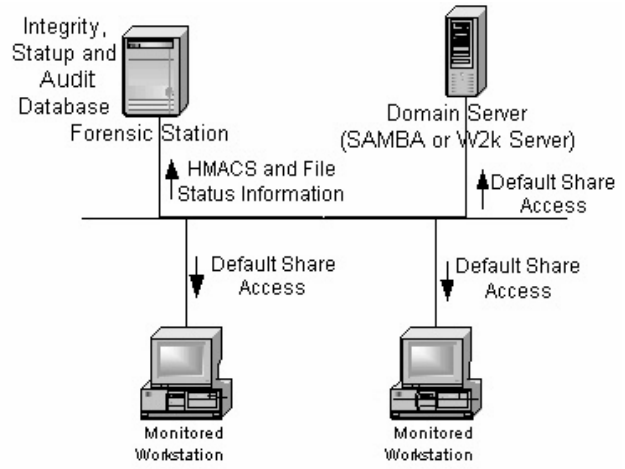

Figure 1: File Integrity Module Schema

The machines to be monitored do not need any special configuration to comply with PFSAF; the only requirement is the existence of the W2k default shares.

Currently the main functionalities of PFSAF are:

- **Guarantee of critical files integrity:** the PFSAF file integrity module generates cryptographic hashes, registers the existence and size of alternate streams and stores the extremely volatile MAC Times (access, modification and creation times of a file);
- **Automatic execution program monitoring:** all programs automatically executed after user logon has its integrity monitored. Moreover, any change in the number of programs to be executed or alterations in their characteristics can be detected;
- **Auditing policy configuration:** PFSAF can remotely configure auditing policies and check whether they have been changed afterwards.

5.2- Implementation

PFSAF was implemented using PERL and it was tested with the ActiveState[5] 5.6.1 Build 631 interpreter on a Windows 2000 Server with Service Pack 2, having Windows 2000 Server and Professional clients on a variety of Service Pack combinations.

There are basically two configuration files in this framework: `files.cfg` and `audity.cfg`. The file `files.cfg` holds filenames to be included in the integrity database (`hash.db`), which is responsible for the storage of cryptographic hashes and some extra data which describe the structure and the state of the target file.

File `files.cfg` is used for all machines listed in file `audit.cfg`, unless there exists a specific file for a given machine, whose name has the following syntax: `<machine>-files.cfg`. In this case, file `files.cfg` is neglected under the more specific configuration listing.

Names of the machines to be monitored are stored in file `audit.cfg`, which also describes which auditing policies must be applied on each listed machine.

In next section will be discussed implementation details of each module presented in Section 5.1.

5.2.1- File Integrity Module

The file integrity module uses both PFSAF configuration files: `files.cfg`, which is specific to this module and `audit.cfg`, which holds the list of machines to be monitored, besides the description of auditing policies for each machine.

Remote access to each machines' files is done through Windows *default shares*, which, though criticized by some, have become extremely useful for system administration.

For cryptographic hashes, PFSAF uses SHA1 paired with the HMAC symmetric key primitive [14], also called HMAC-SHA1. Its use prevents a simple cut-and-paste attack to the `hash.db` file, in which the attacker could substitute a compromised hash for the correct one, therefore validating a possible change in the monitored files.

HMAC-SHA1 uses a key to compute the file hashes, so that it is not possible for anyone not possessing that key to produce any valid hashes for substitution over the `hash.db` file. In the case of PFSAF, a key is required when generating the database and stored in the `passwd` file (also part of PFSAF). The HMAC-SHA1 support in PERL is given by the Digest::HMAC_SHA1 library.

5.2.2- Startup Integrity Database

The Startup Integrity Module is responsible for monitoring and ensuring the integrity of all automatically executed programs after user logon. Its goal is to prevent a maliciously altered program or backdoor from being automatically executed in any of the monitored machines.

There are two ways for a program to automatically execute when a user logs on a Windows machine: the simplest

5. http://www.activestate.com/Products/ActivePerl/

one is through the addition of shortcuts in the Startup folder, present in all user profiles. To deal with this case, PFSAF accesses the folder holding all user profiles, again by using Windows default shares. The shortcut analyses are made easier thanks to Win32::Shortcut, a very helpful PERL library. Once the executable pointed to by the shortcut is known, PFSAF adds two registers to the Startup Database: one holds the hash and the state of the shortcut file and the other the executable's.

The other way is the addition of proper values in some registry keys (eg. HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run). In this case the PFSAF stores information about the keys and about the state of the programs referenced by the values in these keys.

5.2.3- Audit Policy Module

The goal of the audit policy module is to supply a new interface for auditing policies configuration, thus allowing that either PDCs, which do not share domain configurations, isolated machines or Unix servers running SAMBA, may be quickly configured in an automated way.

Through the use of `audit.cfg`, the audit policy module can configure machines listed in it with their corresponding policy. This module can also check the current policy state on each machine and verify that the comply with current values in `audit.cfg`.

Access to this kind of configuration is granted by PERL using the Win32::Lanman library, that works as a wrapper for the Windows LANMAN API.

5.3- Usage

PFSAF can be used either interactively or automatically. In the interactive option, the program uses the `main.pl` script that provides an interface for using the whole framework, with configuration files prepared beforehand.

The non-interactive use of PFSAF can be done calling individual modules, separately developed to supply a flexible interface to the framework and so allowing the execution of independent modules through Windows Task Scheduler.

6- Conclusion

The use of tools like PFSAF can promote the creation of incident response programs by institutions, since their complex deployment on Windows 2000 based networks can be greatly reduced through task automation. Also, it makes basic requirements for this kind of program more apparent, consequently contributing to spread out the necessity of deploying such programs.

References

[1] WYK, Kenneth R. van; FORNO, Richard; *Incident Response*; O'Reilly; 2001;

[2] FARMER, Dan; VENEMA, Wietse; Forensic Computer Analysis: An Introduction; *Dr. Dobb's Journal*; September 2000; http://www.ddj.com/articles/2000/0009/0009f/0009f.htm;

[3] CARVEY, Harlan; System Security Administration for NT; *LISA-NT -- The 3rd Large Installation System Administration of Windows NT Conference*; Seattle, Washington - USA; 2000

[4] AUGUSTO, Alessandro; de GEUS, Paulo L.; GUIMARÃES, Célio C.; Administration of Large Windows NT Network with DoIt4Me; *The 10th International Conference on System Administration, Networking and Security*; Baltimore, MD, USA; May 2001;

[5] Tripwire Open Source's Home Page:
Visited in 10/01/2002; <http://www.tripwire.org>;

[6] OLIVEIRA, Flávio de Souza; GUIMARÃES, Célio Cardoso; REIS, Marcelo; de GEUS, Paulo Lício; Forense Computacional: Aspectos Legais e Padronização; *Anais do Wseg'2001 - I Workshop de Segurança em Sistemas Computacionais*; Florianópolis, SC - Brazil; pp. 80-85 *(in Portuguese);*

[7] NOBLETT, Michael G.; POLLITT, Mark M.; PRESLEY, Lawrence A.; Recovering and Examining Computer Forensic Evidence; *Forensic Science Communications*, Vol. 2 N. 4 October 2000; Federal Bureau of Investigation;

[8] REIS, Marcelo A.; de GEUS, Paulo L.; Standardization of Computer Forensic Protocols and Procedures; *Proc. of the 14th Annual FIRST Computer Security Incident Handling Conference*; Hawaii, USA; June 2002;

[9] BREZINSKI, Dominique; Building a Forensic Toolkit That Will Protect You From Evil Influences; *The Black Hat Briefings '99;* Las Vegas, NV, USA;

[10] OLIVEIRA, Flávio de Souza; GUIMARÃES, Célio Cardoso; REIS, Marcelo; de GEUS, Paulo Lício; Metodologias de Análise Forense para Ambientes Baseados em NTFS; *Anais do SSI'2001 - III Simpósio de Segurança em Informática*; São José dos Campos, SP - Brazil; pp. 83-90 *(in Portuguese);*

[11] MANDIA, Kevin; PROSISE, Chris; *Incident Response: Investigating Computer Crime*; Osborne/McGraw Hill; 2001;

[12] MICROSOFT, Corp.; Securing Windows® 2000 Network Resources; *Microsoft TechNet*; Microsoft Corporation; July 2000;

[13] MICROSOFT, Corp.; *MCSE Training Kit – Microsoft Windows 2000 Active Directory Services*; Microsoft Press; 2000;

[14] MENEZES, Alfred J.; van OORSCHOT, Paul C.; VANSTONE, Scott A.; *Handbook of Applied Cryptography*; CRC Press; 1996;