Mobility-Enhanced File Integrity Analyzer For Networked Environments

Guantong Wang, Ronald F. DeMara, Adam J. Rocke
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
demara@mail.ucf.edu

ABSTRACT

The ability to monitor computer file systems for unauthorized changes is a powerful administrative tool. Ideally this task could be performed remotely under the direction of the administrator to allow on-demand checking, and use of tailorable reporting and exception policies targeted to adjustable groups of network elements. This paper introduces *M-FICA*, a *Mobile File Integrity and Consistency Analyzer* as a prototype to achieve this capability using mobile agents. The M-FICA file tampering detection approach uses MD5 message digests to identify file changes. Two agent types, *Initiator* and *Examiner*, are used to perform file integrity tasks. An *Initiator* travels to client systems, computes a file digest, then stores those digests in a database file located on write-once media. An *Examiner* agent computes a new digest to compare with the original digests in the database file. Changes in digest values indicate that the file contents have been modified. The design and evaluation results for a prototype developed in the *Concordia* agent framework are described.

Keywords:Mobile Agents, File Integrity, Message Digest, and Computer Security

1. INTRODUCTION

On-demand deployment of mobile agents for file integrity checking can address several challenges present in client/server environments. The mobile agents travel between network hosts to inspect changes to the local file system using local system recourses. Upon successful inspection, they return to report the results.

Agents can use a cryptographic hash function applied to a message to generate a digest representing the message. Message digests are similar to checksums or cyclic redundancy checks in that they represent the contents of a message in a relatively short number of characters. There are two important properties of message digest algorithms. The first is that the algorithm cannot be easily reversed. That is, with at least 128 bits of output, a brute force attack has 1.7 x 10E38 possible input values of the same length to evaluate before finding one that generates the correct output. Consequently, it is unlikely that any two different documents produced at random during the course of human history would have the same 128-bit message. The second useful property of message digest algorithms is that a small change in the input results in a significant change in the output.

There are many message-digest functions available today. All of them work in roughly the same way, but they differ in speed and specific features [1]. One of the most widely used message digest functions is the MD5 [2] function, which was developed by Ronald Rivest. The MD2, MD4, and MD5 message digest functions all produce a 128-bit number from a block of text of any length. Each of them pads the text to fixed-block size, and then performs a series of mathematical operations on successive blocks of the input. The MD2 algorithm has relatively few weaknesses, but it is computationally demanding. MD4 was designed to overcome the speed limitation. MD5 was introduced based on potential attacks against MD4 to include one more round of internal operations and several significant algorithmic changes.

2. COMMERCIALLY-AVAILABLE FILE ANALYZERS

The advantages and disadvantages of three popular file integrity checkers, Tripwire, Veracity and Tiger, are presented. The ability of mobile agents to address these disadvantages is described. Finally, the design, implementation and testing of M-FICA is presented.

Tripwire

Tripwire [3] is an integrity-checking program that gives system administrators the ability to monitor file systems for added, deleted, and modified files. A high level model of Tripwire operation [4] is shown in Figure 1. This shows how the Tripwire program uses two inputs: a configuration file describing the file system objects to monitor, and a database of previously generated signatures.

The Tripwire policy file lists the system and data files to monitor as specified by the administrator. When running Tripwire software for the first time, a baseline database of the file system is created from the policy file. Subsequent operation compares the current files against this baseline database to identifies any changes, additions, or deletions. If a policy violation is detected, it will be identified and described in a violation report.

Tripwire software works at the most fundamental layer protecting the servers and workstations that make up the corporate network. Tripwire contains two major packages: Manager and Connector. The Manager is installed on a host machine and acts as the console and reporter. The Connector is installed on each protected client and contains configuration, database, policy, and report files. Features and Benefits of Tripwire are listed in Table 1.

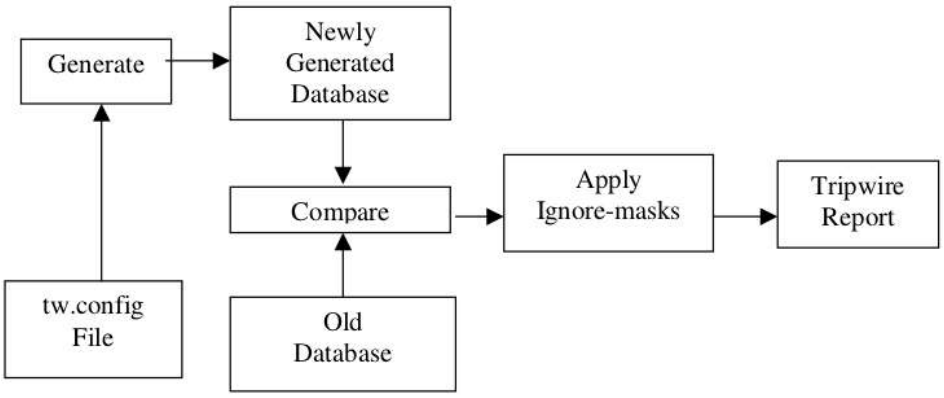

Figure 1: Tripwire Design

Table 1: Features and Benefits of Tripwire

FEATURE	BENEFIT
HQ Manager Compatible-- interface with the enterprise-wide management console	Allows user to have an upgrade path that gives them enterprise-wide control when used with the Tripwire HQ Manager product.
Cryptographic Signing-- database, policy and report files can be cryptographically signed	Reduces the need for removable media for the database, policy, report, or Tripwire data file.
Reporting-- report violations at configurable levels of detail.	Different levels of reports allow users to select a response to Tripwire violations.
Email Reporting-- violation reports can be emailed to a recipient.	Send reports via email to the appropriate systems administrator(s) based upon individual rule violations.
Enhanced Policy Language-- a series of rules that specify Tripwire software checks the integrity of the system.	The policy language has been expanded to include flexibility in defining rules, as well as the ability to prioritize violations based upon severity.
Severity Rating-- files can be given separate severity levels.	Prioritize critical system files with a higher severity level. Reports that have a "high" severity rating can be selected for immediate viewing.

While Tripwire is a popular file integrity checker, there are some disadvantages as described below. These disadvantages of Tripwire are addressed by deploying mobile agent technology:

- *Lack of Interoperability*. Tripwire has limited flexibility in updating the system for new attacks.
- *Extensive network usage*. Tripwire processes most of this data locally. Data is sent to remote network locations where the data is further abstracted, and then eventually sent to a central processing site that evaluates results from all location in the network.
- *Expensive to install*. Tripwire scheme assumes that a host-based checker is installed on every host.
- *Single point of Failure*. Because of Tripwire's central processing system, a failure may disable the entire security system.

Veracity

Veracity [5] is a computer program that detects changes in file systems by creating and manipulating snapshots of directory trees. A snapshot consists of an ordinary text file that records the structure of the directory tree, the names of the files in the tree, and the cryptographic digests of the files in the tree. This snapshot can later be used to generate a list of changes in the tree since the snapshot was made. Client/server networking allows an administrator to monitor the integrity of hundreds of computers from a single point. Table 2 shows the comparison of Tripwire and Veracity. Other Veracity characteristics are listed below:

- *Taking A Snapshot*. The snapshot appears as a file in the directory at the root of the tree.
- *Checking Trees*. A snapshot file can be used to detect changes that have been made in the directory tree since the snapshot was taken. This process, repeated regularly, enables the snapshot to track the changing file system while providing an unbroken chain of integrity checking.
- *Snapshots*. Snapshot files are ordinary text files that store the structure of the directory tree they represent, along with the names and cryptographic hashes of the files in the tree.
- *Snapshot files* are at most 70 characters wide and contain only printable ASCII characters.

Table 2: Comparison of Tripwire and Veracity

	Tripwire	**Veracity**
Architecture	Client / Server	Client / Server
Language	Standard C with micro header file	Funnelweb: a literate-programming macro preprocessor
Available for many platforms	Yes	Yes
Able to check both files and directories	Yes - The configuration file contains a list of entries enumerating the set of directories or files to be monitored	Yes - It takes a snapshot of an entire directory tree recording the tree structure, file names, and cryptographic digests of each file.
Solve the false-positive problem: able to ignore files that frequently change	Yes - Use selected mask that describes which file attributes can be changed without being reported as an exception	Yes - Use a flexible means for specifying which subtrees or files are to be checked and to what extent.
Full strength cryptographic hashes	Yes	Yes
Able to monitor entire network from a single point	Yes - Tripwire HQ Manager. This is a software "console" that enables control of HQ Connectors across an network from a central location.	Yes - Veracity's snaplets system simplifies this automated checking allowing monitoring of an entire network.
Officially allocated port	Unknown	The Internet Assigned Numbers Authority has assigned TCP port 1062 for use by Veracity/FreeVeracity.
Resource limits	Unknown	The server configuration file allows limits to be placed on the server's use of processes, memory and (using access delays) CPU time.
Cryptographic Signing	Yes - The Tripwire database, policy, and optional report file can be cryptographically signed	Yes
Violation reports can be emailed to specified recipients.	Yes	Yes
Multiple Levels of Reports	Users can choose from five levels of reports.	Unknown

Tiger

Tiger [6] is a set of scripts that search a system for weakness which could allow an unauthorized user to change system configurations, gain root access, or change important system files. Included in the package are a static audit tool, a signature database to check system binaries against known signatures of patch files, and a network traffic analyzer that aids system administrators in assessing outside threats. Tiger scans the *cron, inetd,* and *passwd* files for vulnerabilities. It also inspects file permissions, aliases, and *PATH* variables to see if they can be used to gain root access [7]. Scans for system vulnerabilities in *inetd*, host equivalents, and *PATH* variables are performed to determine if a user can gain remote access to the system. MD5 signatures are used to determine if key system binaries have been altered. Tiger scripts may be run together or individually to allow system administrators to determine the best strategy for checking their system

Most available security tools fails into two categories: static audit tools and integrity checkers. The foundations of integrity checking programs, such as Tripwire or Veracity, are that a database is created with some unique identifier for each file to be monitored. By recreating that identifier, which could be a copy of the entire file contents, and comparing it against the saved version, it is possible to determine if a file has been altered. Furthermore, by comparing entries in the database, it is possible to determine if files have been added or deleted from the system.

Tiger serves as a static audit tool. It is a checklist program, similar to the shell scripts. A checklist is one form of this database for a UNIX system. The file contents themselves are not usually saved, as this would require too much disk space. Instead, a checklist would contain a set of values generated from the original file - usually including the length, time of last modification, and owner. The checklist is periodically regenerated and compared against the saved copies to identify discrepancies from the stored values. A particular weakness of Tiger is that a user gaining access to the root account may modify the raw disk to alter the saved data without it showing in the checklist.

3. ROLE OF MOBILE AGENTS IN FILE INTEGRITY

Mobile agents are program instances that are able to move within a network under their own control. Mobile agents consist of code, a data state, and an execution state. Mobile agents are able to autonomously migrate, communicate to each other, and offer services or interfaces to applications.

Characteristics of Mobile Agents

Mobile agents serve as a framework on top of which decentralized infrastructure services can be built. By embedding functionality in mobile software agents distributed across the network, intelligence traditionally centralized in a few controlling nodes is pushed out into the system at large. The following are important characteristics for mobile agents:

- encapsulate a thread of execution and preserve data when it moves from one network node to another,
- move easily across the network,
- must be small in size due to the cost associated with hosting and transporting an agent,
- are able to cooperate with other agents in order to perform complex or dynamic tasks, and
- are able to identify and use resources specific to any node.

Mobile Agents for File Integrity

There are several motivations for consideration of mobile agents: [8] [9] [10]

- *Mitigate network latency*. For critical real-time systems, latencies are not acceptable. Multiple agents offer a solution as they can be dispatched from a central controller to act locally and directly execute the controller's directions.
- *Reduce network load*. Mobile agents reduce the flow of raw data in the network as they move the computations to the data rather than the data of the computations.
- *Execute asynchronously and autonomously*. Tasks embedded in mobile agents become independent of the creating process and can operate asynchronously and autonomously.
- *Adapt more dynamically*. Mobile agents also have the ability to sense their execution environment and autonomously react to changes.
- *They are naturally heterogeneous*. Because mobile agents are generally computer- and transport-layer-independent and are dependent only on their execution environment, they provide optimal conditions for seamless system integration.
- *Provide robust and fault-tolerant behavio*r. The ability of mobile agents to react dynamically to unfavorable situations enables building of robust and fault-tolerant distributed systems.

4. MOBILITY-ENHANCED NETWORK FILE INTEGRITY (M-FICA) PROTOTYPE

The Mobile File Integrity and Consistency Analyzer, M-FICA, is a file integrity checker using mobile agent techniques. M-FICA agents are sent from a host to a client machine, reside there to check file changes of the local system using local system recourses, then return to report the results.

Functional Requirements

There are two basic routines to complete file integrity tasks. The baseline database is created by computing a digest for each file designated in the policy file. Subsequent scans compute a new digest for each monitored file and compare the new result with the baseline.

Environmental Requirements

In a distributed computing environment, it is common for several platforms to exist. Therefore, Java was selected as programming language because of its platform independence. M-FICA agents are programmed using the Concordia mobile agent API. It is necessary that the system can be run on any hardware platform. Therefore, any machine that can offer a Java 1.1 run-time environment can meet the basic needs of the system.

Design Issues and Features

Issue 1. Because of the heterogeneous nature of computer equipment at most sites, the code is written in Java language.

Issue 2. Detecting file tampering by comparing each file against a duplicate copy requires considerable storage and time. Generating and comparing file signatures may require more computation, but it requires much less storage so the signature approach is selected.

Issue 3. Message digests represent the contents of a message in a relatively short number of characters. MD5 is selected as the encoding algorithm to digest the file contents.

Issue 4. Agents shall be small, specialized pieces of code. For maximum efficiency, two agents are designed to complete the task: Initiator and Examiner. The size of all six associated classes is less than 5K bytes.

Issue 5. Policy file and baseline database file integrity is critical. To avoid being corrupted, those files are stored on removable media.

High-level Design

The Mobile Agent File Integrity Analyzer project uses two agents: Initiator and Examiner to complete the functional requirements. Initiator, illustrated in Figure 2, reads the configuration, or policy, file and then reads each designated file from the local system. It then computes a digest for each file and reports to the host. All the digests are collected in a baseline file stored on read-only media.

Examiner, illustrated in Figure 3, reads the policy and system files, then generates new message digests using the MD5 algorithm. It then reads original message digests from baseline and compares each new message digest with baseline. Finally it produces a list of changed files to report to the host.

5. EXPERIMENTAL RESULTS

A test file is created to evaluate agent operation. Initiator is launched to compute a baseline digest.

Examiner then computes another digest for comparison. The results shall be the same when file contents are unmodified. For subsequent tests, the test file is modified. When Examiner checks the file again, the results shall be changed. The result is illustrated in Figure 4.

Result shows that the Initiator/Examiner agent protocol is able to identify changes in file contents resulting in an incorrect message digest. This can provide an effective means to manage file integrity checking in dynamic environments where numerous instances of policy files dictating which files should be scanned are needed to be maintained on the remote hosts. Instead this information is dispatched with the Examiner agent when needed from the administrator's site. While current capabilities of M-FICA include reliable detection of file modifications, the policy administrator console remains as future work.

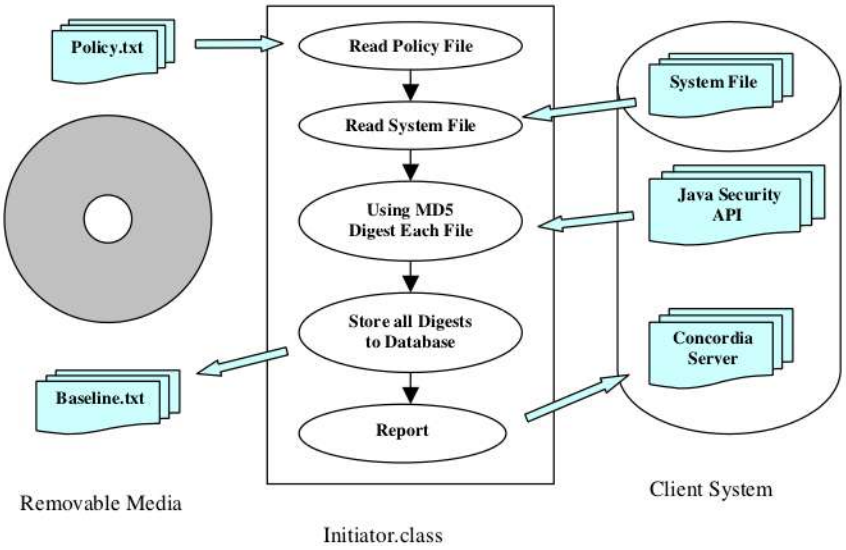

Figure 2: Mobile agent Initiator architecture

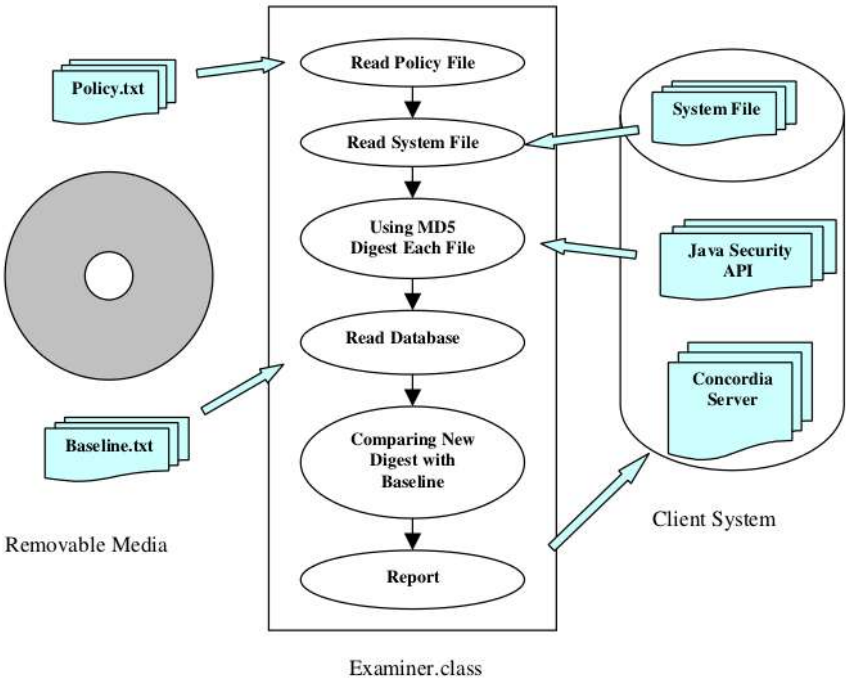

Figure 3: Mobile agent Examiner architecture

6. CONCLUSION AND FUTURE WORK

The Mobile File Integrity and Consistency Analyzer (M-FICA) prototype has been designed and assessed. Results show that deploying two agent types can form a sufficient protocol for file integrity checking. Initiator and Examiner agents perform the file integrity tasks while maintaining space-efficient operation requiring as little as 5 KB/agent of Java code. Even when under conditions of network load, results show agents of that size to keep detection delay well below 5 seconds.

Furthermore, deploying Java agents overcomes the heterogeneous environment barrier. The test indicated that the same software could also be used in Windows 98 and Windows NT environments in multiple network labs without modification. Thus, agent-based integrity checkers can be cost effective when compared with client/server-based architectures. Agent-based integrity checkers can provide more readily-maintainable security tools that can be more directly customized on a dynamic basis upon deployment.

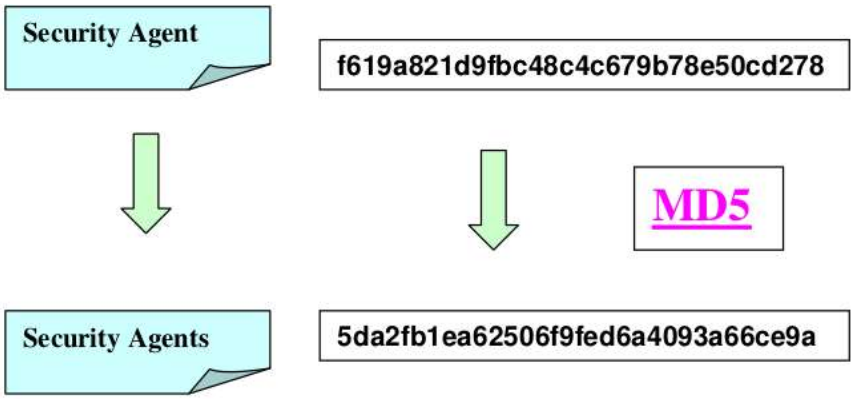

Figure 4: Change detected for file containing the text 'Security Agent'.

Future research tasks include developing a centralized control console and semi-automatic adjustments to the policy file by prompting the administrator for updates to address the false positive problem inherent in all integrity analyzers. Also, use of different levels of security measurement by the agents is to be explored, along with experiments to determine Examiner agent size and complexity limits based on network loading and latency trade-offs.

LIST OF REFRENCES

[1] Bruce Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, Inc., 1996.

[2] R. L. Rivest. RFC 1321: *The MD5 Message Digest Algorithm*. Technical Report, Internet Activities Board, April 1992.

[3] Gene H. Kim, Eugene H. Spafford, *The Design and Implementation of Tripwire: A File System Integrity Checker*, COAST Laboratory, Department of Computer Sciences, Purdue University, 1995. <URL: ftp://coast.cs.purdue.edu/pub/papers/gene>

[4] Gene H. Kim, Eugene H. Spafford, *Experiences with tripwire: Using integrity checks for intrusion detection.* In System Administration, Networking and Security Conference III, Usenix, 1994.

[5] Rocksoft Limited, 1999, *Veracity Network Integrity*, < URL: http://www.veracity.com/index.shtml>

[6] Daniel Framer, Eugene H. Spafford, *The Cops Security Check System TAMU Security Tools - Tiger,* 1993, <URL: http://www.net.tamu.edu/network/tools/tiger.html >

[7] Bobby S. Wen. *Open-Source Intrusion-Detection Tools for Linux*, Linux Journal, October 2000.

[8] Simon Y. Foo and Michael Arradondo, *Mobile Agents for Computer Intrusion Detection*, Proceedings of the 36th Southeastern Symposium on System Theory, March 2004.

[9] Wayne Jansen, Peter Mell, Tom Karygiannis, Don Marks, *Applying Mobile Agents to Intrusion Detection and Response*, National Institute of Standards and Technology, Computer Security Division, 1999.

[10] Oleg Kachirski and Ratan Guha, *Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks*, Proceedings of the 36th Annual Hawaii International Conference on System Sciences, January 2003.