Evaluation of Distributed File Integrity Analyzers in the Presence of Tampering*

Adam J. Rocke[1], Ronald F. DeMara[1], and Simon Foo[2]
(Corresponding author: Adam J. Rocke)
Department of Electrical and Computer Engineering, University of Central Florida Orlando[1]
4000 Central Florida Blvd., Orlando, FL 32816–2450, USA
Department of Electrical and Computer Engineering[2]
The Florida A&M University and the Florida State University, Tallahassee, FL 32310–6046, USA

(Received Oct. 11, 2005; revised and accepted Dec. 3, 2005 & Feb. 6, 2006)

Abstract

In this paper, the Collaborative Object Notification Framework for Insider Defense using Autonomous Network Transactions (CONFIDANT) is evaluated in the presence of tampering. CONFIDANT's mitigation capabilities are assessed and compared with conventional file integrity analyzers such as AIDE and tripwire. The potential of distributed techniques to address certain tampering modes such as Pacing, Altering Internal Data, and File Juggling are discussed. To assess capabilities, a variably-weighted tampering mode exposure metric scheme is developed and utilized. Results indicate a range of vulnerabilities for which mitigation techniques such as Encapsulation, Redundancy, Scrambling, and mandatory obsolescence can increase robustness against challenging exposures, including various insider tampering risks.

Keywords: File system integrity, intrusion detection evaluation, network-level security, tampering exposures, weighted metric evaluation scheme

1 Introduction

An Intrusion Detection System (IDS) serves to identify breaches of computer system security and performs an important role in protecting computer systems from tampering [3, 6, 7, 8]. File integrity analyzers are a class of related tools that verify the correctness of security-critical files in order to detect suspicious modification.

Two popular tools are tripwire and the Advanced Intrusion Detection Environment (AIDE). Tripwire utilizes a policy file to describe permitted changes to critical system files. A baseline database is created by applying cryptographic hash functions to system files as specified in the policy file. Future scan results are compared to entries in the baseline database and a report containing the status of system file contents is generated. AIDE and other tools operate similarly.

Existing frameworks reviewed in [1], including Tripwire and AIDE, exhibit a single point-of-failure and are subject to insider tampering. The Collaborative Object Notification Framework for Insider Defense using Autonomous Network Transactions (CONFIDANT) aims at trusted detection of unauthorized modifications to filesystem data. The design of CONFIDANT is based on two goals in order to limit exposures present in existing frameworks. These goals are:

Goal-1: to reduce single point-of-failure exposures and

Goal-2: increase barriers against insider tampering.

The tampering mode vulnerabilities defined in [1] include spoofing, termination, sidetracking, altering internal data, and selective deception categories. The CONFIDANT design approach to mitigate these exposures, including user capability classification, is provided in [9]. The CONFIDANT agent framework utilizes four autonomous *behaviors* operating in three distinct echelons as defined in [10]. An IDS taxonomy, example handshaking scenarios, and CONFIDANT evaluation results for Goal-1 are also provided therein. Related agents, a^i_j, communicate within committees, C^i, where i is the committee index and j is the committee agent ID. Interlocked agent communication conveys file scan results, network status, and ensures that failure of an individual agent or network node is detected by other committee members. Results show that CONFIDANT does not exhibit the single point-of-failure present in existing frameworks.

In this paper, the CONFIDANT file integrity verification framework is evaluated with respect to robustness in the presence of tampering. Evaluation in the presence of insider tampering for each tampering mode is provided in Section 2. A comparative metric weighting scheme is

*Supported in-part by National Security Agency subcontract MDA904-99-C-2642.

Table 1: Numerical measures for the file integrity problem

Numerical Measure	Description
True Positive (TP)	Modification of a monitored file followed by an appropriate alarm
False Positive (FP)	The presence of an alarm when no file modification has occurred
True Negative (TN)	The absence of both a file modification and an alarm
False Negative (FN)	A file modification without an associated alarm
Sensitivity (Sen)	Probability that a file modification is identified when present ($\frac{TP}{TP+FN}$)
Specificity (Spec)	Probability that an alarm is not sounded when a file modification is not present ($\frac{TN}{FP+TN}$)

proposed in Section 3 to evaluate performance relative to the existing frameworks Tripwire [4, 11] and AIDE [5].

2 Evaluation Methodology and Results

As listed in Table 1, an alarm generated in response to an authorized file modification is a FP. Leveraging the fact that any alarm might potentially provide useful intrusion information, every file modification encountered by CONFIDANT agents that is not permitted is reported and considered a TP. An alarm generated by an agent due to network outages may be considered a FP if it is the result of benign activity. The presence of an alarm generated by such activity is considered to be a TP, as network or gateway failure may be an indication of malicious intent where an insider tries to circumvent CONFIDANT agent interactions. In this case, alarms are generated in order to notify administrators of the unavailability of these network resources.

Evaluation in the presence of insider tampering is provided below. Tripwire and AIDE are tested in addition to CONFIDANT for comparison. The default Tripwire and AIDE configuration performs integrity scans once daily following the NIST recommendation for file integrity scan intervals [12]. In order to facilitate testing, scan timing is increased to once every minute. Tripwire and AIDE scan results are provided in an email message to the administrator to serve as alarm notification, while CONFIDANT uses communication and interlocking between agents to provide alert information.

2.1 Testing of Spoofing Tampering Modes

Transmitting counterfeit data in order to mislead the recipient is Spoofing. Spoonfeeding sensor information not present in the target file, Sugarcoating unfavorable reports prior to evaluation, and Recanting alert notification are all instances of Spoofing. CONFIDANT is designed to mitigate exposures to Spoofing tampering modes by employing techniques such as direct interaction between the agent gateway and hardware resources on local hosts, SSL communication between remote hosts, and agent interlocking.

All existing IDS frameworks including CONFIDANT operate at the application layer. As a result, several non-trivial but potentially serious vulnerabilities remain. CONFIDANT mitigation techniques are based on the assurance of an initial known safe state, robust agent communication, and gateway integrity. For this to be the case, the hardware and operating system kernels must remain free of tampering. Techniques to mitigate tampering at the hardware and operating system level are the subject of future research.

2.2 Testing of Termination Tampering Modes

Test Case: TC-Blindfolding

Blindfolding an IDS involves disabling sensor processes. Since the Tripwire and AIDE processes perform sensor and control routines, termination of the scanning process is tampering by Blindfolding. Termination of the IDS process causes a race condition where if the verification process was able to complete operations before scan operation was disabled, the scan was allowed to inspect file contents. Knowledge of scan times increases the viability of Blindfolding.

The `cron` daemon emails any process execution output as a report to the task owner. Termination of the Tripwire process results in an alarm stating that execution terminated due to an uncaught signal in place of the normal scan result. Termination of the AIDE process results in the absence of an AIDE-specific response, but `cron` reports that the process exited due to an uncaught signal. In this case, the expected alarm is never observed, but insight into potential tampering is provided. One problem is that the response is generated by `cron` and not AIDE. It is possible that an administrator, by expecting an AIDE-specific response, will ignore the `cron`-generated notification. Since the Tripwire and AIDE processes are spawned by `cron`, another Blindfolding technique is to terminate the `cron` daemon. Termination of `cron` results in the absence of a response. This is expected because if the scan is never initiated, no response can be generated.

Termination of the CONFIDANT gateway process results in the destination being unavailable during an attempted agent dispatch. When a gateway is unavailable, an alarm notification is sent to other committee members. Similarly, when file integrity scan results are unavailable, as is the case when agent a_j^i is terminated, other commit-

Table 2: Termination test results

	Tripwire	**AIDE**	**CONFIDANT**
No Tampering	Cron Output (TP)	AIDE Output (TP)	Alarm (TP)
Terminate IDS Process	Cron Error Output (TP)	Cron Error Output (TP)	Remote Alarm (TP)
Terminate Mail Process	Cron Error Output (TP)	Cron Error Output (TP)	Alarm (TP)
Terminate Cron Daemon	No Alarm (FN)	No Alarm (FN)	Alarm (TP)
Sensitivity	0.725	0.725	1
Specificity	1	1	1

tee agents $a^i_{k\neq j}$ trigger an alarm.

A random number generator was used to select between the absence of tampering, termination of the IDS, mail, or `cron` processes over many tests. Results are listed in Table 2. Of the three tested frameworks, CONFIDANT is most resistant to Blindfolding as an alarm is generated if a scan cannot be performed. In the best case, termination of Tripwire and AIDE processes will report that an error has occurred. In the worst case, termination of the `cron` daemon causes Tripwire and AIDE to fail completely. Termination of a CONFIDANT gateway prompts remote committee agents to report that a resource is unavailable as described previously.

Test Case: TC-Commandeering

Due to sensor and control functions being performed within a single process, the testing of Commandeering follows the same steps as Blindfolding, and has the same result listed in Table 2. Here again those processes are terminated prior to generation of alarms. Tripwire and AIDE either report error messages or have no response at all, while CONFIDANT generates an alarm as the scan cannot be performed.

Test Case: TC-Soundproofing

Soundproofing involves disabling of alarm components. Termination of email response results in a single notification for both Tripwire and AIDE. Table 2 lists the response in the presence of disabling email services. Since Tripwire utilizes `cron` for email alarms, termination of email services has no effect on response. The response from AIDE, however, does not exist in the same manner as described in TC-Blindfolding. The AIDE-specific response cannot be generated, and output is handled by `cron`. Termination of the mail process, even when the file under inspection is not modified, results in a message provided by `cron` stating that "`aide has returned many errors.`" Termination of `cron` as described in TC-Blindfolding results in no response of any kind from either Tripwire or AIDE.

Since CONFIDANT does not rely on email for alarm notification, disabling of email services had no effect on alarm operation. Disabling of alarm components involves termination of a gateway on a specified host. Such behavior is recognized either by committee members executing on remote nodes or agents that attempt to travel to the terminated gateway as described previously.

2.3 Testing of Sidetracking Tampering Modes

Test Case: TC-Blockading:

Blockading involves isolating a sensor from needed access to a monitored file or device. One blockading technique is to increase the system load using a high priority process in order to prevent the request for a filesystem scan from being serviced. Testing involves using the `stress` program [13] to impose load on a computer system including CPU, I/O, virtual memory, and disk stress. The maximum sustained load was approximately 45 as reported by the `uptime` system utility. In this test case, system load was gradually increased to the maximum sustained load. During this time, Tripwire and AIDE were scheduled to perform scans every minute while CONFIDANT agents traversed the network.

Results follow the general trend of increased delay under increased system load. The greatest delay encountered is 198 seconds while under a system load of 25.02 as listed in Table 3. While Tripwire and AIDE continued to generate reports under loads of 25-45, reports arrived out of order. Based upon execution order, AIDE reports are expected to appear first followed by Tripwire reports. Delays exceeding 60 seconds result in a group of AIDE reports followed by a group of Tripwire reports. Also, in approximately 30% of reports generated while the load was above 25, Tripwire is unable to complete the scan operation. CONFIDANT agents signal connection alarms as host services could not be accessed in a timely manner under any load imposed by `stress`.

Blockading of IDSs with network components can also be performed by increasing network load to forestall access. The benefits of mobile agents described in [10] include imposing minimal execution overhead, communication cost reduction, and a reduced network load compared to traditional client-server techniques. In order to investigate CONFIDANT network performance, agents are dispatched in the presence of an increasing network load as described below.

In order to measure agent network performance, traffic

Table 3: Blockading and pacing test results

Tampering Mode	IDS		
	Tripwire	AIDE	CONFIDANT
Blockading	TP delayed up to 198 sec	TP delayed up to 193 sec	TP
Pacing	FN	FN	TP

is generated at a defined rate while agents traverse the network. Traffic is captured using `tcpdump` and written to a file. The traffic is replayed using `tcpreplay` with the rate parameter, `-r`, to specify the rate in megabits per second (Mbps) and the topspeed parameter, `-R`, to replay the traffic at the maximum rate. The total round-trip dispatch and acknowledgment time for the agent to travel one hop in the network is recorded. Since agent network travel increases the overall network load, traffic rate is obtained using `tcpstat` reporting one second intervals. For each specified load, an agent performs repeated round-trip cycles, and the average traversal time was obtained as listed in Table 4.

As network traffic increases, the ability of agents to traverse the network is diminished. Network load up to approximately 20 Mbps results in a traversal time between 215 and 300 msec. The travel time increases significantly when load increases past 20 Mbps. Figure 1 illustrates the agent traversal time in the presence of increasing network load. Agent traversal times are relatively consistent up to a network load of 20 Mbps. High network loads greater than 20 Mbps steadily increase traversal delays. Under maximum network load of nearly 30 Mbps, agents are still able to traverse between gateways in under 700 msec. The interlocking nature of CONFIDANT agents requires that successful tampering occur at multiple gateways. Blockading increases agent dispatch delays and, thus, increases the time during which tampering can occur. For a network with n hops between the tampered node and the alarm destination, detection of a remote intrusion arrives within time δ by using an exponential spreading notification scheme where $\delta \geq (700msec) \log_2 n$.

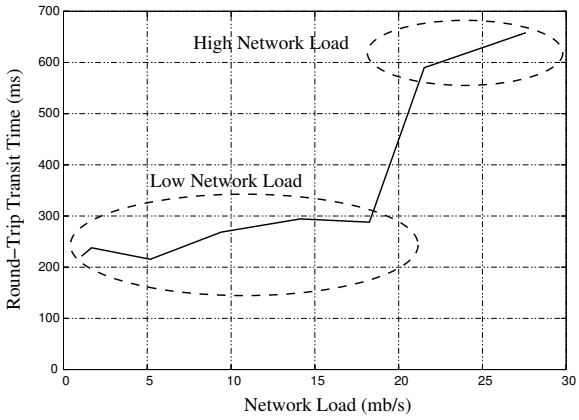

Figure 1: Agent network performance

CONFIDANT mitigates Blockading by taking network latency into consideration. Agent network performance testing provides accurate estimates of latency values based on the physical network topology and expected load. A delay in excess of an expected value is a potential sign of tampering as the remote gateway may be unreachable. Examples of Blockading network access include DoS attacks and physically disabling hardware resources. If an agent is unable to travel to the destination gateway within the expected time period, an alert is generated to inform security or administrative personnel of potential tampering.

The maximum observed traversal time of any single round-trip dispatch iteration under all tested network loads was 3.46 seconds. Based on this performance testing, it can be seen that the allowed delay should be at least 3.5 seconds to take into consideration agent delays due to routine network traffic as shown in Figure 1.

Test Case: TC-Pacing

Pacing involves tampering with external timing mechanisms. For tools that perform scans at times specified by the system clock, tampering can be performed by resetting the system clock. Figure 2 illustrates tampering by Pacing with a file integrity scan time of t_{scan} and time intervals τ. Initially the time-of-day clock and the actual time are the same, more formally $t = t_{TOD}$. When t enters the period $t_{scan} - \tau < t_{TOD} < t_{scan}$, the interval immediately preceding the file system scan, it is set to $t_{TOD} = t_{TOD} + 2 * \tau$ effectively skipping past the scan. Now the actual time appears to be in the interval $t_{scan} + \tau < t_{TOD} < t_{scan} + 2 * \tau$. When two τ periods have elapsed, t_{TOD} will be in the interval $t_{TOD} > t_{scan} + 3 * \tau$. The time-of-day clock is then reset to the actual time, $t_{TOD} = t_{TOD} - 2 * \tau = t$, file system scan has been bypassed, and the system clock has been restored to the actual time.

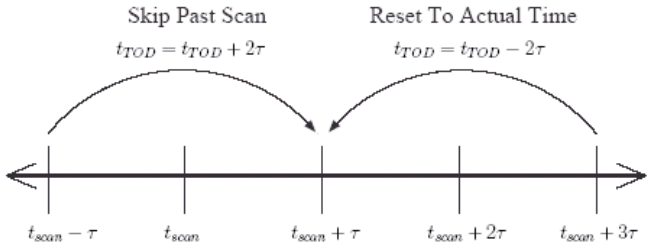

Figure 2: Pacing timeline

Table 4: Network load and agent traversal time

Load Specified by tcpreplay (Mbps)	Load Observed by tcpstat (Mbps)	Average One-hop Round-trip Agent Traversal Time (msec)	Maximum Round-trip Traversal Time (sec)
0	1.074652	221.33	0.383
1	1.677341	237.89	0.416
5	5.183018	215.55	0.249
10	9.410184	268.55	0.447
15	14.135512	294.33	0.491
20	18.255930	287.88	0.457
25	21.529032	590.00	3.463
Maximum	27.588796	658.11	3.241

Tripwire and AIDE scans occur at periodic intervals based on the system clock and are consequently vulnerable to Pacing. During Pacing tests, Tripwire and AIDE were not able to make a single TP detection. The fact that Tripwire and AIDE are subject to pacing by reliance on the `cron` system utility is an implementation detail. Even if scan timing information is stored internally by Tripwire and AIDE, integrity scans remain periodic and are both predictable and configurable. In both cases, they are vulnerable to tampering via Pacing. CONFIDANT agents do not rely on the system clock to trigger any event and filesystem scans are not scheduled to occur at fixed times. Agent scan timing is a function of the travel itinerary and corroboration with committee members. By employing internal scan timing mechanisms that do not depend on the system clock, CONFIDANT was able to detect all file modifications with ideal sensitivity and specificity and is not vulnerable to Pacing as listed in Table 3. Attempts to modify the scan timing mechanism is considered altering internal data as described below.

Test Case: TC-Scapegoating

Triggering alarms with the intent of overwhelming the alarm subsystem is Scapegoating. One technique involves artificially increasing the false alarm count in order to divert the attention of security personnel away from the tampering. Another Scapegoating technique is to produce additional alarm messages to consume all available disk space thus denying further local alerts to be stored. Scapegoating can be performed by writing multiple alarms to disk or by creating a process to generate additional alarms as listed in Table 5. The main concern with Scapegoating is that alarm messages must be processed by the human administrator to verify their validity. Multiple alarms are generated in order to overload the administrator. An advantage that all file integrity tools have compared to network intrusion detection systems is the number of file scans, and consequently the number of potential alarms should be relatively low. Based on the default operation, Tripwire and AIDE should produce a single report per day. The presence of hundreds of alarms, even without consideration for the content of the alarm, is a sign of an error at some level and indicates tampering via Scapegoating.

Tripwire and AIDE are subject to tampering via Scapegoating, as the result of file integrity scans are provided in email form. Simple text processing can be used to insert erroneous messages into a security administrator's inbox. The current version of CONFIDANT employs messages displayed on the local console as well as messages transmitted to agents. It is possible to enable a process to present alarm messages on gateway consoles. For instance, errors detected by a local agent are relayed to remote agents within the committee, so the absence of corresponding alerts on remote nodes is an indication of Scapegoating. Also, multiple committee agents $a_1^1 \ldots a_n^1$ visit the node in question so alarms will be confirmed by multiple agents. Alarm messages from only one agent a_i^1 without corroboration from other agents in committee C^1 within the specified time window are indicative of Scapegoating.

Table 5 lists various technique considerations in message-centric and process-centric categories. In order to tamper by Scapegoating, an insider can either create a process to generate an alarm or create a process to perform intrusions resulting in alarms. Message-centric techniques only require knowledge of alarm format while process-centric tampering requires knowledge of an existing vulnerability. This helps process-centric techniques to appear to be more convincing than message-centric tampering. Due to the reliance on email for alarm notification, Tripwire and AIDE are particularly susceptible to message-centric techniques while CONFIDANT is not. Since alarm messages are distributed to remote nodes, successful Scapegoating in CONFIDANT requires significant additional effort to distribute appropriate erroneous alarm messages. Process-centric techniques, however, perform local intrusions and rely on agent interlocking to distribute alarm data to remote gateways. This increases confidence as alarms are generated in response to an actual intrusion.

2.4 Testing of Altering Internal Data Tampering Modes

Tampering with internal data involves after-the-fact modification of an IDS component that is completely installed and properly configured prior to misuse. Unlike previous tampering modes, altering internal data tampering interacts directly with the IDS without termination of any of its logical components.

Test Case: TC-Retroactive Baselining

Table 5: Scapegoating technique considerations

Mode of Attack	Message-Centric	Process-Centric
	Produce message data corresponding to additional alarms	Create a process to perform intrusive activity
Attacker Knowledge Required	Format of alarm messages	Existence of a vulnerability
Plausibility to Observer	Low to moderate	High
Tripwire Susceptibility	High	High
AIDE Susceptibility	High	High
CONFIDANT Susceptibility	Low	High

Table 6: Effort and outcome estimates for tampering via altering internal data

Tampering Mode	IDS		
	Tripwire	**AIDE**	**CONFIDANT**
Retroactive Baselining	Single operation < 5 minutes effort 10 of 10 resulted in FN	Single operation < 5 minutes effort 10 of 10 resulted in FN	Multiple operations > 8 hours effort 10 of 10 TP detected by remote agents
Descoping	Single operation < 5 minutes effort 10 of 10 resulted in FN	Single operation < 5 minutes effort 10 of 10 resulted in FN	Multiple operations > 8 hours effort 10 of 10 TP detected by remote agents
Value Jamming	Single operation < 5 minutes effort 10 of 10 resulted in FN	Single operation < 5 minutes effort 10 of 10 resulted in FN	Multiple operations > 8 hours effort untested

File integrity tools compute a baseline value for monitored files when the host is in a safe state for comparison with future integrity scans. Tripwire and AIDE store baseline values in a local database file. Baseline data in CONFIDANT is internal to the mobile agents. Tampering by Retroactive Baselining involves modification of the baseline value. Both Tripwire and AIDE provide mechanisms to reinitialize or update the local baseline database. Each enables an insider to reinitialize the local database and reconcile differences between the existing database and current file state.

Reinitializing the local Tripwire database updates the hash values of the modified files. With updated baseline data, Tripwire is unable to recognize tampering with the monitored file. Tripwire documentation states that the baseline database should be stored on read-only media to mitigate tampering by outsiders. A read-only baseline database does not mitigate insider risk as an insider with physical access could replace or reconfigure the media. Updating the baseline database results in a False Negative as modifications are not detected. Creating a new AIDE database then copying it to the default location is a method used by administrators to reflect approved file hash values. Retroactive Baselining of Tripwire and AIDE is a system manageability issue. Enabling administrators to update the baseline database provides the defined method to facilitate insider tampering. Once baseline data is modified, no valid tampering detections were made and a False Negative is encountered.

In order to tamper with CONFIDANT agents via Retroactive Baselining, the baseline data contained within the agent must be modified while in memory. Due to the dynamic nature of CONFIDANT agents, successful tampering requires that an attacker must:

- physically locate every agent within a committee across the network,
- determine the baseline memory location in each agent on the local and remote hosts, and
- update the memory location for all agents between message exchange and prior to dispatch.

In order to test the CONFIDANT response to Retroactive Baselining, internal modifications are simulated as illustrated in Figure 3. Two agents, a_1^1 and a_3^1, perform filesystem scans and post an event to agent a_2^1. The baseline value in a_1^1 is valid while the value in a_3^1 has been tampered with. The simulation involves the modification being present prior to initial dispatch as opposed to altered while executing. Both agents a_1^1 and a_3^1 send the message: $send-msg.(a_2^1,\ MD5\_OK)$ to agent a_2^1 stating that the scan result is negative. The internal baseline MD5 value is passed as part of the `MD5_OK` event. Agent a_2^1 detects a discrepancy between its internal baseline and the one from agent a_3^1. An alarm is triggered and dispatched to all members of committee C^1. File modification is detected using propagation of alarm notification as illustrated in [10].

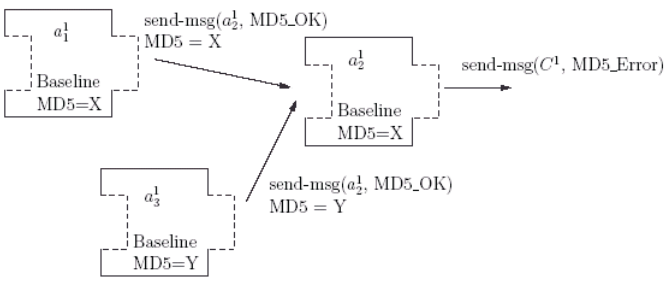

Figure 3: Agent interaction in the presence of retroactive baselining

If only a single agent a_3^1 is subjected to tampering,

other agents within the committee C^1 will detect the baseline discrepancy and generate an alarm. An adversary must simultaneously determine the memory location of every agent within a committee on distributed nodes in the monitored network. Testing is simulated by having modified baseline data contained within one committee agent upon initial dispatch. While this agent considers a modified file to be valid, interlocking messages between other committee members results in an alarm due to incongruent baseline data between committee members.

Test Case: TC-Descoping

Descoping differs from Retroactive Baselining in that integrity scan policy is modified as opposed to the scan baseline values. File integrity scan policy data specifies the files to be scanned.

In order to modify policy data in Tripwire, a policy database file must first be modified and the baseline database must then be reinitialized. Descoping in AIDE is performed by editing the configuration file and reinitializing the database. Removing policy file data will prevent the file in question from being scanned. A cognizant observer notices that a file to be monitored is missing from the scan report as the email notification lists the files that were scanned. In the presence a large scan with many entries, such information may be overlooked.

As with the baseline data, all CONFIDANT agent policy information is stored internally. The same steps required to tamper by Retroactive Baselining are involved to tamper by Descoping with the exception of modifying policy as opposed to baseline data. Tripwire and AIDE specifically allow an administrator to reconfigure baseline and policy information thereby allowing tampering by an insider. CONFIDANT has no such mechanism to update baseline or policy information.

Descoping testing is performed by simulation of modified policy data contained within one committee agent upon initial dispatch. As in the Retroactive Baselining test, two agents perform filesystem scans and post an event. Here the baseline value in a_3^1 is `null` as the policy data is modified to omit the scan. In this case agent a_3^1 does not send a message as no scan is performed. Since a message is expected but not received, agent a_2^1 sends the message: $send-msg.(C^1,\ MD5\_Error)$ to other members of committee C^1 to acknowledge that tampering has occurred.

Test Case: TC-ValueJamming

Value Jamming involves altering internal data in some way so that alarms are ignored. One technique is to continuously write `FALSE` to a status location in memory as to indefinitely delay alarm notification. A less involved technique specifically for tools that employ email as the alarm mechanism is to modify or delete email contents. Once Tripwire and AIDE have delivered email messages detailing the result of the daily file integrity scan, the email can either be modified to reflect that file integrity is intact or it can be replaced with a copy of a previous message with updated header information. Successful tampering will effectively eliminate any alarms.

Value Jamming in CONFIDANT employed the same steps listed in TC-Retroactive Baselining to modify memory locations to disable agent messages. Messages in CONFIDANT serve as both communication and alarm notification. In this case, the CONFIDANT response to Value Jamming is the same as the response for Descoping. When messaging is disabled a message is expected but not received, thus activating the propagation of alarm notification.

Value Jamming in CONFIDANT can also be performed by continuously asserting an internal memory modification so that scan messages always send a `MD5_OK` event. This is transmitted with the expected MD5 value, even if the internal baseline data is invalid. Successful Value Jamming in a single agent results in passing valid scan messages without regard for the results of the scan. This prevents alarm messages from being generated. Subsequent gateway visits by other agents in C^1 provided alarm notification as they remained unmodified. A successful adversary must simultaneously tamper with each agent in committee C^1. Testing of both continued modification of individual committee C^1 agents on subsequent visits, and simultaneous modification of all agents in committee C^1, was unsuccessful. Thus, Value Jamming is mitigated in CONFIDANT by employing spatially and temporally distributed agents. Multiple agent visits on each gateway utilize a range of memory addresses. Also, multiple agents may reside on a gateway simultaneously. These efforts prevent tampering by modifying of a single memory location from being successful.

2.5 Testing of Selective Deception Tampering Modes

The ability to accurately predict integrity scan intervals is required to perform undetected tampering by File Juggling. Consider a scan time of t_{scan}, and a time period τ. Once t_{scan} is determined, operations can be performed before and after the scan in order to hide tampering. A file integrity scanner is susceptible to tampering by File Juggling if a pre-scan operation at time $t_{scan}-\tau$ and post-scan operation at time $t_{scan}+\tau$ can successfully hide file modifications.

Test Case: TC-FileJuggling

File Juggling is performed by executing pre-scan operations to present filesystem data in the valid state in conjunction with post-scan attack operations. For the file integrity problem, these operations are copying valid data to the scan location prior to the scan, then replacing it with the modified data after the scan, as shown below. Tripwire and AIDE scan times are readily determined by inspection of the `cron` daemon configuration. File Juggling is illustrated in Figure 4. Scans are scheduled to occur at time t_{scan_1} and t_{scan_2}. At time $t_{scan}-\tau$, the valid file is available for scan operations. At time $t_{scan}+\tau$, the

Table 7: TME metric weighting scheme

Category	Name	Weight	Rationale
Management	Monitoring	4	Security personnel may not remain in a single location
	Configurability	2	Ease of configuration enables insider tampering
	Scalability	2	The test network contains few nodes
Spoofing	Spoonfeeding	2	Attack requires intricate
	Sugarcoating	2	OS-level modification
	Recanting	1	Relies on human administrator response
Termination	Blindfolding	3	Trivial attack pathway
	Commandeering	3	for any insider
	Soundproofing	3	
Sidetracking	Blockading	1	Successful attempts delay accurate results
	Pacing	2	Modifying scan timing can prevent scan from occurring
	Scapegoating	1	Relies on human administrator response
Alter Internal Data	Retroactive Baselining	4	Baseline changes can make tampered data appear to be valid
	Descoping	4	Policy changes can make tampered data appear to be valid
	Value Jamming	4	Eliminating alarms gives a false sense of security
Selective Deception	File Juggling	3	Predictable scan timing facilitates future tampering

valid file is replaced with a maliciously altered version. For testing purposes, scans are performed every five minutes with the interval τ set at one minute. Due to the use of periodic scan intervals, Tripwire and AIDE were unable to detect file tampering as the valid file was presented during scan operations. Since CONFIDANT does not rely on the system clock for timing information, scans are neither regularly scheduled or predictable. CONFIDANT was able to detect file modifications with perfect sensitivity. It is important that an agent can begin the scan operation and obtain filesystem data prior to an operating system context switch. If an attacker can monitor a process list, detect process initialization, and perform operations prior to the agents obtaining filesystem data, modified data can be replaced with valid data prior to MD5 hash computation.

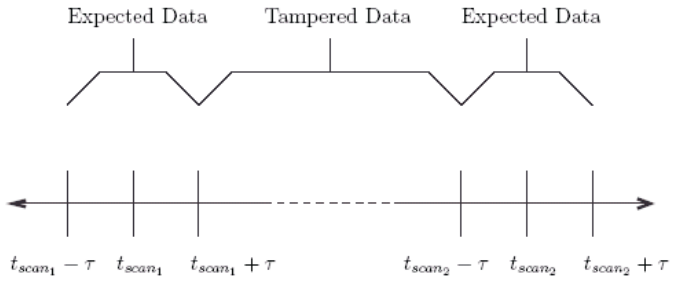

Figure 4: Expected and tampered data presented during file juggling

Valid data is observed by the IDS during the intervals $t_{scan} - \tau < t < t_{scan} + \tau$. During the interval $t_{scan_1} + \tau < t < t_{scan_2} - \tau$, filesystem data has been tampered with. The probability of the filesystem data being in the expected valid state is:

$$p_v = \frac{(t_{scan_1} + \tau) - (t_{scan_1} - \tau)}{t_{scan_2} - t_{scan_1}} = \frac{2\tau}{\Delta t_{scan}}.$$

Increasing τ decreases the probability of the data being in a modified state. When Δt_{scan} is set to 24 hours per NIST guidelines, $p_v = 0.0069$ even if the scan takes as long as 5 minutes.

3 TME Weighting Scheme and IDS Comparison

A metric weighting model called the Tampering Mode Exposure (TME) weighting scheme is developed based on the metric evaluation strategy described in [2]. In order to compare the frameworks numerically, categories and weights are defined and results computed using Equation (1) with j categories, $i = n$ metrics in each category j, and an unweighted score U_{ij} and weight W_{ij} for each metric. Six categories, $j = 6$, are defined including one for each of the five tampering mode classes and a management category. The assigned weights and rationale for weight selection are listed in Table 7. Weights are given values of 1 to 4 based on the relative significance of each metric based on the methodology in [2]. Higher values indicate greater capability for management metrics and increased significance of successful tampering for tampering mode metric classes.

$$S = \sum_{j=1,6} \left[\sum_{i=1,n} (U_{ij} * W_{ij}) \right]. \quad (1)$$

Unweighted scores are listed in Table 8. Scores are assigned a value of 1, 2, or 3 to signify detection failure, a modified result, and correct operation, respectively, for the tampering mode classes. For instance, testing of Selective Deception resulted in Tripwire and AIDE generating false negatives, so they are assigned a score of 1. CONFIDANT provided accurate alarm notification and is assigned a score of 3. Scores of the management category

metrics are assigned based on the individual significance of each exposure.

Monitoring specifies the ability to receive alarm notification from multiple locations. The distributed nature of CONFIDANT provides alarms across the monitored network domain and is assigned a score of 3. AIDE has no network capability and is assigned a score of 1. The use of Tripwire Manager allows alarms to be received at a central console, thus providing greater monitoring ability than AIDE, but not fully distributed as in CONFIDANT. Configurability as it relates to insider tampering is discussed in the previous section. Tripwire and AIDE utilize configuration files that can be modified by an administrator and are assigned high scores. A CONFIDANT design consideration is to disallow configuration to eliminate certain insider tampering exposures.

Using the weights in Table 7 and the scores in Table 8, a comparison of the frameworks can be performed. The weighted results calculated using Equation (1) are 65, 59, and 103 for Tripwire, AIDE, and CONFIDANT respectively, out of a maximum score of 123. CONFIDANT compares favorably under the TME weighted model where Tripwire and AIDE score comparably to each other. The scores and weights of the TME model, and the categories it uses, can be adapted to evaluate the performance of other IDSs in a similar manner.

4 Conclusion

Testing was performed to illustrate the defined mitigation techniques. Tripwire and AIDE were evaluated in order to compare results with CONFIDANT's response. In the absence of tampering, all frameworks operate correctly. Results from tampering via Recanting, Scapegoating, and to some degree Value Jamming are similar among frameworks, as all rely on security administrator reaction to the presented alarm notification. Blockading causes all three frameworks to warn that resources are unavailable. Tripwire and AIDE reports arrive later than expected and out of order. Furthermore, termination-based tampering causes Tripwire and AIDE to fail completely, while CONFIDANT generates accurate alarm notification. CONFIDANT may be subject to tampering if an adversary is able to simultaneously modify all agents within a committee across a network domain. Attempts to perform such tasks have proven unsuccessful. Across all tampering modes, testing has shown that the CONFIDANT response is at least as accurate as the Tripwire and AIDE response to the same stimulus.

Testing has identified critical exposures in all evaluated frameworks. Tripwire and AIDE exhibit critical exposures to tampering via Pacing, all Altering Internal Data tampering modes, and File Juggling. Every test case for these tampering modes resulted in a FN response. Testing of CONFIDANT shows that it is highly subject to Blockading but carries less significance than those for Tripwire and AIDE. In fact, even under minimal system load, filesystem scans failed, and alarms were generated stating that the scan could not be performed.

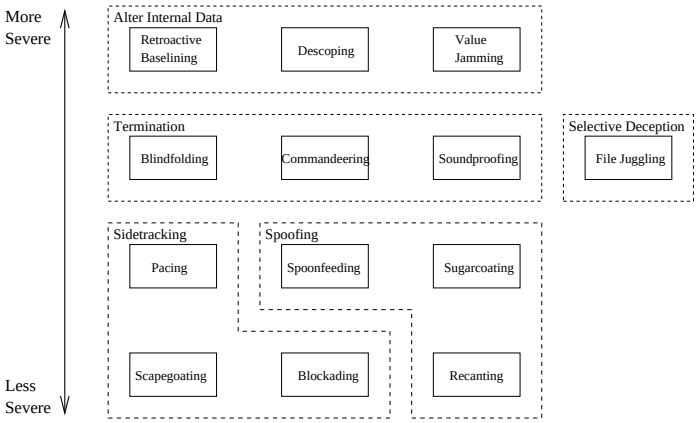

Figure 5: Relative tampering mode impact

While each framework is subject to certain critical exposures, the severity of the associated tampering modes varies, as illustrated in Figure 5. The TME weights listed in Table 7 are based on tampering mode severity. For instance, tampering via Blockading is not as severe as tampering via Pacing. Blockading causes results to be delayed while Pacing has the potential to completely bypass scan operations. Similarly, Pacing is not as severe as Retroactive Baselining as updating the baseline database causes the IDS to interpret all results as valid, while configuration and scan timing remains unchanged and therefore undetected. Testing has shown that the Altering Internal Data tampering modes can be the most severe, while Scapegoating, Blockading, and Recanting are not as detrimental nor effective.

Testing also showed the relationship between configurability and robustness against insider tampering. Specifically, testing has shown CONFIDANT to be effective in mitigating several severe insider tampering exposures at the expense of manageability. IDSs that provide configuration and management routines inherently enable insider tampering. This can best be seen by inspection of the test results for the Altering Internal Data tampering modes. Consider Retroactive Baselining in Tripwire as opposed to CONFIDANT. Tripwire includes commands to allow an administrator to update baseline data. Once the data has been updated with a MD5 of a modified file, subsequent scans report that the file is valid.

The most significant difference between existing approaches and CONFIDANT with regard to manageability is that CONFIDANT does not readily allow incremental upgrade of the IDS during network operation. While conventional frameworks may allow administrator-approved updates during operation, the CONFIDANT approach would interpret such modifications as a form of insider tampering. Thus, an administrator is unable to easily perform tasks such as updating the internal baseline data as CONFIDANT does not have built in management routines. When upgrades are required the entire IDS func-

Table 8: TME unweighted scores

Metric	Tripwire	AIDE	CONFIDANT	Rationale
Monitoring	2	1	3	Alarm notification on multiple nodes
Configurability	3	3	1	Ability to reconfigure once deployed
Scalability	2	1	3	Overlapping agents vs centralized control
Spoonfeeding	1	1	1	Architectural vulnerability between OS and application
Sugarcoating	1	1	1	
Recanting	3	3	3	Administrator response
Blindfolding	2	2	3	Test result listed in Table 2
Commandeering	2	2	3	
Soundproofing	2	2	3	
Blockading	2	2	1	Test result listed in Table 3
Pacing	1	1	3	
Scapegoating	2	2	2	Administrator response
Retroactive Baselining	1	1	3	Test result listed in Table 6
Descoping	1	1	3	
Value Jamming	1	1	3	
File Juggling	1	1	3	Reliance on system clock

tionality must be temporarily disabled prior to upgrading. The TME management category aims at quantifying the tradeoff between robustness in the presence of insider tampering and manageability. Administrators can adjust TME weights to determine the impact for each implementation instance.

A distributed design is essential for robust operation in the presence of insider tampering as it enforces successful tampering to occur at multiple nodes simultaneously. Two major difficulties with this design include the interlocking of distributed components and recovery upon intrusion detection. Agent interlocking is required to ensure that components remain distributed. Also, since manageability is sacrificed in order to enhance robustness against insider tampering, recovery after alarm notification required that CONFIDANT be restarted on all nodes. This may not be practical for large enterprise installations.

While the current work focused on individual tampering modes, future work includes investigation of cascading tampering modes as certain tampering modes may be combined to increase IDS exposures. For instance, the susceptibility of an IDS to Selective Deception may be increased with resource blockades and high priority processes. Tampering by Pacing or Blockading may allow File Juggling to occur as illustrated in Figure 6. Successful File Juggling depends on the predictability of scan timing. An attacker could first perform Blockading to delay IDS access to filesystem resources and then perform File Juggling resulting in a successful attack. Similarly, Pacing can be performed to corrupt the system time by setting the system clock to a time when a scan is known to not occur in order to facilitate File Juggling. Another example involves tampering via Blockading in order to localize agents to a smaller network domain than defined upon initial dispatch. This may increase the exposure to Termination tampering modes.

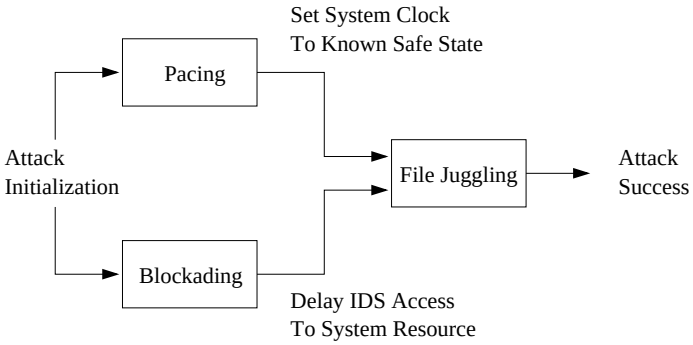

Figure 6: Cascading tampering mode pathway example

References

[1] R. F. DeMara and A. J. Rocke, "Mitigation of network tampering using dynamic dispatch of mobile agents", *Computers & Security*, vol. 23, no. 1, pp. 31-42, 2004.

[2] G. Fink, B. Chappell, T. Turner, and K. O'Donoghue, "A metrics-based approach to intrusion detection system evaluation for distributed real-time systems", in *proceedings of the 16th International Parallel and Distributed Processing Symposium*, pp. 93-100, Fort Lauderdale, FL, USA, 2002.

[3] Y. Jianga, Z. Xiab, and S. Zhanga, "A novel defense model for dynamic topology network based on mobile agent", *Microprocessors and Microsystems* vol.29, pp. 289-297, 2005.

[4] G. H. Kim and E. H. Spafford, "Experiences with tripwire: Using integrity checkers for intrusion detection", in *proceedings of the 3rd Annual System Administration, Networking and Security COnference*, pp. 89-101, Toronto, Ontario, Canada, 1994.

[5] R. Lehti, "Advanced intrusion detection environment". http://www.cs.tut.fi/*sim*rammer/aide.html

[6] D. G. Marks, P. Mell, and M. Stinson, "Optimizing the scalability of network intrusion detection systems using mobile agents", *Journal of Network and Systems Management* vol, 12, no. 1, pp. 98-110, 2004.

[7] J. McHugh, A. Christie, and J. Allen, "Defending yourself: The role of intrusion detection systems", *IEEE Software*, vol. 17, no. 5, pp. 42-51, 2000.

[8] J. McHugh, A. Christie, and J.Allen, "Intrusion detection: Implementation and operational issues", *Software Engineering Institute Computer Emergency Response Team White Paper*, 2001. http://www.stsc.hill.af.mil/crosstalk/2001/01/mchugh.html

[9] A. J. Rocke and R. F. DeMara, "Mitigation of insider risks using distributed agent detection, filtering, and signaling", *International Journal of Network Security*, vol. 2, no. 2, pp. 141-149, 2006.

[10] A. J. Rocke, and R. F. DeMara, "Collaborative object notification framework for insider defense using autonomous network transactions", *Autonomous Agents and Multi-Agent Systems* vol. 12, no. 1, pp. 93-114, 2006.

[11] Tripwire, Inc. Tripwire.org-home of the tripwire open source project. http://www.tripwire.org

[12] J. Wack, M. Tracy, and M. Souppaya, "Guideline on network security testing", *National Institute of Standards and Technology, Computer Security Division, Special Publication* SP-800-42, Oct. 2003. http://csrc.ncsl.nist.gov/publications/nistpubs/800-42/NIST-SP800-42.pdf

[13] A. Waterland, "Stress: impose a configurable amount of computer system load". http://weather.ou.edu/∼apw/projects/stress/

Adam J. Rocke received the Ph.D. degree in Computer Engineering from the University of Central Florida in 2004. He has been a lecturer and researcher at the University of Central Florida in the areas of system software and network security. Dr. Rocke's research interests are in distributed processing, network security, and embedded systems.

Ronald F. DeMara received the Ph.D. degree in Computer Engineering from the University of Southern California in 1992. Since 1993, he has been a full-time faculty member at the University of Central Florida and is currently a Professor with joint appointment in the Departments of Electrical and Computer Engineering and Computer Science. Dr. DeMara's research interests are in Distributed Processing and Special-Purpose Architectures. He is the editor of two books and has approximately 100 publications in-print or online on these topics. His research has been sponsored by the National Science Foundation, NASA, U.S. Army and Navy, National Security Agency, Harris Computer Systems, Lockheed Martin Information Systems, Theseus Logic Incorporated and others. He is a Senior Member of IEEE and a Member of ACM and ASEE. He has served on the Editorial Boards of the Journal of Circuits, Systems, and Computers, the journal Microprocessors and Microsystems, and IEEE Transactions on VLSI Systems.

Simon Y. Foo is a Professor at the Department of Electrical and Computer Engineering at Florida State University. His research interests are in neural networks, genetic algorithms, fuzzy logic, machine vision, and hardware field programmable gate array (FPGA) implementations. Dr. Foo has authored or co-authored more than seventy refereed technical papers on these topics. He also graduated more than 25 graduate students. His primary sponsors include the National Security Agency, National Science Foundation, U. S. Air Force, Boeing Aircraft Company, and the Florida Department of Transportation. In addition to the prestigious "Engineering Research Award" (2004) from the FAMU-FSU College of Engineering, he also won the "Teacher of the Year" award (2001) two "Best Paper" awards (2001 and 2004), and the Teaching Incentive Program Award (1995).