# A Note on Detecting Tampering with Audit Trails

by Dr. Frederick B. Cohen ‡

In this paper, we examine methods for detecting attempts to subvert the integrity of audit trails.

Search terms: IT Audit, Tamper Detection



‡ This research was funded by *Management Analytics*, PO Box 1480, Hudson, OH 44236, USA
Url http://all.net - Email fc@all.net - Tel 1+216-686-0090 - Fax 1+216-686-0092

# 1 Background and Introduction

In a recent hearing, the content and meaning of audit trails of a possible attack were at issue. During the hearing, the issue came up of whether the audit trails may have been tampered with. A prosecution witness claimed that the attacker might have tampered with the audit trails. When asked whether there was any evidence of such tampering, the prosecution witness could not support the contention with any evidence. In the end, the audit trails presented were found not to be indicative of a malicious attack, in part because they prosecution could not produce any evidence of tampering with the audit trails.

In a general sense, if an attacker gains unlimited access to a system, if audit trails are not protected by write-only or write-once technology, and if no physical means are used of effective in determining the authenticity of audit trails, it is always possible to create a forged audit trail that is undifferentiable from a legitimate audit trail. All an attacker has to do is create a complete audit trails for an equivalent system and replace the real audit trail with the replacement while acting as a privileged user. This can be done with direct hardware input and output, thus avoiding audits of the forgery process itself.

Experience shows that attackers rarely meet the theoretical limits of how effectively they can carry out an attack. Instead, when they try to cover up attacks, they tend to do one of three things. They (1) attempt to delete all files on a system to remove all trace of their entry, (2) try to modify select audit trails to remove the indications of their use, or (3) try to prevent their attacks from being audited by avoiding the use of audited events.

If they attempt to delete all files on a system to remove all trace of their entry, they will either succeed or fail. If they fail, the audit trail of their attack will remain, while if they succeed, we will detect their success by virtue of the resulting disruption. We may or may not be able to recover other details.

If they prevent their attacks from being audited by avoiding the use of audited events, there is little that can be done to detect their tampering within the system.

This paper is about ways to detect tampering in the case where the attacker tries to modify select audit trails to remove the indications of their use. For the purposes of this paper, we will be using examples from a Unix-based system, however, most of the results apply equally well to other multi-user, multi-processing operating environments.

We begin by examining ways in which audit trails may be modified. We then consider the loss of expected behaviors and the introduction of unexpected behaviors that may result from tampering. Next, we look at redundancy in audit trails and how they may be used to detect tampering. Finally, we summarize, draw conclusions, and suggest further work.

# 2 How Audit Trails are Generated and Modified

Most audit trails are generated on a record-by-record basis. Whenever an auditable event occurs, a record is appended to a file containing an audit trail. There may be several files containing different classes of auditable events. Most audit entries include a time and date indicator, an indicator of the object that caused the audit trail, a user identity of the subject associated with the auditable event, and optional information such as file names, error indicators, etc. Here are some example entries from one such audit trail:

```
Oct 23 03:45:46 all in.thttpd[10709]: twist csman.csie.ntu.edu.tw to /usr/etc/in.thttpd csman.csie.ntu.edu.tw unknown
Oct 23 03:46:06 all in.thttpd[10736]: twist csman.csie.ntu.edu.tw to /usr/etc/in.thttpd csman.csie.ntu.edu.tw unknown
Oct 23 03:56:51 all in.fingerd[12339]: connect from 146.227.1.1
Oct 23 03:57:25 all in.thttpd[12415]: twist csman.csie.ntu.edu.tw to /usr/etc/in.thttpd csman.csie.ntu.edu.tw unknown
Oct 23 04:03:08 all sendmail[13076]: AA29270: to=postmaster@liv1.spiretech.com, delay=06:11:38,
      stat=Deferred: Connection refused by liv1.spiretech.com
Oct 23 04:04:14 all in.thttpd[13202]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:04:17 all in.thttpd[13207]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:04:42 all in.thttpd[13259]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:05:04 all in.thttpd[13311]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:14:37 all sendmail[14406]: AA14406: message-id=<199510230156.UAA09239@osh1.datasync.com>
Oct 23 04:14:37 all sendmail[14406]: AA14406: from=<owner-cypherpunks@toad.com>, size=2109, class=-60
Oct 23 04:14:48 all in.thttpd[14421]: twist ad36-131.compuserve.com to /usr/etc/in.thttpd ad36-131.compuserve.com unknown
Oct 23 04:14:52 all in.thttpd[14434]: twist ad36-131.compuserve.com to /usr/etc/in.thttpd ad36-131.compuserve.com unknown
Oct 23 04:15:22 all in.thttpd[14490]: twist ad36-131.compuserve.com to /usr/etc/in.thttpd ad36-131.compuserve.com unknown
Oct 23 04:15:41 all sendmail[14543]: AA14406: to=<fc@all.net>, delay=00:01:06, stat=Sent
Oct 23 04:25:12 all in.thttpd[15635]: twist csman.csie.ntu.edu.tw to /usr/etc/in.thttpd csman.csie.ntu.edu.tw unknown
Oct 23 04:36:22 all in.redirect[16902]: connect from i81m2.ira.uka.de
Oct 23 04:36:44 all in.thttpd[16979]: twist i81m2.ira.uka.de to /usr/etc/in.thttpd i81m2.ira.uka.de unknown
Oct 23 04:36:50 all in.thttpd[16983]: twist i81m2.ira.uka.de to /usr/etc/in.thttpd i81m2.ira.uka.de unknown
Oct 23 04:43:24 all in.thttpd[17732]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:43:52 all in.thttpd[17808]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:47:07 all in.thttpd[18197]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:47:22 all in.thttpd[18224]: twist erik.cica.es to /usr/etc/in.thttpd erik.cica.es unknown
Oct 23 04:54:59 all in.thttpd[19209]: twist t037.is.co.za to /usr/etc/in.thttpd t037.is.co.za unknown
Oct 23 04:55:04 all in.thttpd[19218]: twist t037.is.co.za to /usr/etc/in.thttpd t037.is.co.za unknown
Oct 23 04:55:33 all in.thttpd[19301]: twist t037.is.co.za to /usr/etc/in.thttpd t037.is.co.za unknown
Oct 23 04:56:26 all in.thttpd[19485]: twist t037.is.co.za to /usr/etc/in.thttpd t037.is.co.za unknown
Oct 23 04:57:41 all in.thttpd[19723]: twist t037.is.co.za to /usr/etc/in.thttpd t037.is.co.za unknown
Oct 23 05:03:10 all sendmail[20572]: AA29270: to=postmaster@liv1.spiretech.com, delay=07:11:40,
      stat=Deferred: Connection refused by liv1.spiretech.com
Oct 23 05:12:22 all sendmail[21678]: AA21678: message-id=<Pine.LNX.3.91.951023161515.27409A-100000@ima.net>
Oct 23 05:12:22 all sendmail[21678]: AA21678: from=<owner-cypherpunks@toad.com>, size=2682, class=-60
Oct 23 05:14:38 all sendmail[21935]: AA21935: message-id=<9510230914.AA21935@all.net>
Oct 23 05:14:38 all sendmail[21935]: AA21935: from=fc, size=10212, class=0
Oct 23 05:14:48 all sendmail[21937]: AA21935: to=todd@lgt.com, delay=00:00:10, stat=Sent
Oct 23 05:19:54 all sendmail[22548]: AA21678: to=<fc@all.net>, delay=00:07:34, stat=Sent
Oct 23 05:43:10 all sendmail[25218]: AA25218: message-id=<Pine.SOL.3.91.951023192908.6735B-100000@orb>
Oct 23 05:43:10 all sendmail[25218]: AA25218: from=<owner-cypherpunks@toad.com>, size=1480, class=-60
```

```
Oct 23 05:48:42 all sendmail[25853]: AA25218: to=<fc@all.net>, delay=00:05:34, stat=Sent
Oct 23 05:58:30 all in.thttpd[26973]: twist frank.bfsec.bt.co.uk to /usr/etc/in.thttpd frank.bfsec.bt.co.uk unknown
Oct 23 05:58:42 all in.thttpd[27000]: twist frank.bfsec.bt.co.uk to /usr/etc/in.thttpd frank.bfsec.bt.co.uk unknown
Oct 23 05:58:50 all in.thttpd[27021]: twist frank.bfsec.bt.co.uk to /usr/etc/in.thttpd frank.bfsec.bt.co.uk unknown
Oct 23 05:59:05 all in.thttpd[27054]: twist frank.bfsec.bt.co.uk to /usr/etc/in.thttpd frank.bfsec.bt.co.uk unknown
Oct 23 05:59:17 all in.thttpd[27084]: twist frank.bfsec.bt.co.uk to /usr/etc/in.thttpd frank.bfsec.bt.co.uk unknown
Oct 23 06:02:42 all in.thttpd[27464]: twist 164.11.100.10 to /usr/etc/in.thttpd 164.11.100.10 unknown
Oct 23 06:03:07 all in.thttpd[27542]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
Oct 23 06:03:09 all sendmail[27545]: AA29270: to=postmaster@liv1.spiretech.com, delay=08:11:39,
       stat=Deferred: Connection refused by liv1.spiretech.com
Oct 23 06:03:31 all in.thttpd[27572]: twist gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk unknown
Oct 23 06:03:35 all in.thttpd[27573]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
Oct 23 06:04:10 all in.thttpd[27638]: twist gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk unknown
Oct 23 06:07:12 all sendmail[28011]: AA28011: message-id=<Pine.SUN.3.91.951023022123.9641C-100000@eskimo.com>
Oct 23 06:07:12 all sendmail[28011]: AA28011: from=<owner-cypherpunks@toad.com>, size=2075, class=-60
Oct 23 06:14:15 all sendmail[28822]: AA28011: to=<fc@all.net>, delay=00:07:05, stat=Sent
Oct 23 06:18:42 all in.thttpd[29305]: twist dell.netcraft.co.uk to /usr/etc/in.thttpd dell.netcraft.co.uk unknown
```

In this particular audit trail (a *syslog* file) the date and time information is followed by the networked computer generating the record (e.g., all), the program generating the record (e.g., in.thttpd, sendmail), the process number associated with the record (e.g., [27572]) and the output generated by the auditable event (e.g. twist...).

Audit trails are typically generated by a number of sources. For example, the program *in.thttpd* generates its own audit trail:

```
csman.csie.ntu.edu.tw unknown 1995/10/23 03:45:46 cat /searcher/top.html
csman.csie.ntu.edu.tw unknown 1995/10/23 03:46:07 cat /searcher/searchhint.html
csman.csie.ntu.edu.tw unknown 1995/10/23 03:57:26 cat //books/orange/appendc.html
erik.cica.es unknown 1995/10/23 04:04:14 cat /index.html
erik.cica.es unknown 1995/10/23 04:04:17 cat /allnet.gif
erik.cica.es unknown 1995/10/23 04:04:42 cat /integ/top.html
erik.cica.es unknown 1995/10/23 04:05:04 cat /searcher/top.html
ad36-131.compuserve.com unknown 1995/10/23 04:14:48 cat /index.html
ad36-131.compuserve.com unknown 1995/10/23 04:14:52 cat /allnet.gif
ad36-131.compuserve.com unknown 1995/10/23 04:15:23 cat /browse.html
csman.csie.ntu.edu.tw unknown 1995/10/23 04:25:13 cat //books/orange/appendc.html
i81m2.ira.uka.de unknown 1995/10/23 04:36:45 cat /index.html
i81m2.ira.uka.de unknown 1995/10/23 04:36:50 cat /allnet.gif
erik.cica.es unknown 1995/10/23 04:43:24 cat /integ/top.html
erik.cica.es unknown 1995/10/23 04:43:53 cat /browse.html
erik.cica.es unknown 1995/10/23 04:47:08 cat /help.html
erik.cica.es unknown 1995/10/23 04:47:22 cat /index.html
t037.is.co.za unknown 1995/10/23 04:54:59 cat /index.html
```

```
t037.is.co.za unknown 1995/10/23 04:55:05 cat /allnet.gif
t037.is.co.za unknown 1995/10/23 04:55:33 cat /admin/mlist/top.html
t037.is.co.za unknown 1995/10/23 04:56:29 cat /admin/mlist/9510.html
t037.is.co.za unknown 1995/10/23 04:57:41 cat /admin/mlist/9509.html
frank.bfsec.bt.co.uk unknown 1995/10/23 05:58:30 cat /index.html
frank.bfsec.bt.co.uk unknown 1995/10/23 05:59:17 cat /allnet.gif
164.11.100.10 unknown 1995/10/23 06:02:42 cat /index.html
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:08 cat /heaven.html
gate.uwe.ac.uk unknown 1995/10/23 06:03:32 cat /searcher/top.html
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:35 cat /admin/mlist/top.html
dell.netcraft.co.uk unknown 1995/10/23 06:18:42 cat /searcher/top.html
```

This audit trail covers the same time period on the same system as the earlier audit trail, but includes detailing of the operation performed (e.g., cat) and the file on which that operation was performed (e.g., /searcher/top.html).

Another audit trail details every process executed by giving the program name, special characteristics, the user executing the process, the controlling terminal (if any), the runtime of the process, and the date and time of the event. This audit trail commonly produces several entries per second.

```
sync            root     ttyp1      0.20 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.08 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
sh              fc       ttyp0      0.03 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.06 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.05 secs Sat Oct 21 07:01
cat             fc       ttyp0      0.06 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.05 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.05 secs Sat Oct 21 07:01
sed             fc       ttyp0      0.08 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.05 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.05 secs Sat Oct 21 07:01
```

```
tail            fc      ttyp0     0.05 secs Sat Oct 21 07:01
tail            fc      ttyp0     0.06 secs Sat Oct 21 07:01
tail            fc      ttyp0     0.06 secs Sat Oct 21 07:01
tail            fc      ttyp0     0.08 secs Sat Oct 21 07:01
sh        F     fc      ttyp0     0.02 secs Sat Oct 21 07:01
sh        F     fc      ttyp0     0.02 secs Sat Oct 21 07:01
ls              fc      ttyp0     0.06 secs Sat Oct 21 07:01
date            fc      ttyp0     0.08 secs Sat Oct 21 07:01
sync            root    ttyp1     0.17 secs Sat Oct 21 07:01
sync            root    ttyp1     0.16 secs Sat Oct 21 07:01
ls              root    ttyp1     0.06 secs Sat Oct 21 07:01
sync            root    ttyp1     0.16 secs Sat Oct 21 07:01
sync            root    ttyp1     0.17 secs Sat Oct 21 07:01
sync            root    ttyp1     0.17 secs Sat Oct 21 07:01
sync            root    ttyp1     0.22 secs Sat Oct 21 07:01
sync            root    ttyp1     0.20 secs Sat Oct 21 07:01
grep            fc      ttyp0     0.05 secs Sat Oct 21 07:01
grep            fc      ttyp0     0.06 secs Sat Oct 21 07:01
sh              fc      ttyp0     0.02 secs Sat Oct 21 07:01
tail            fc      ttyp0     0.09 secs Sat Oct 21 07:01
grep            fc      ttyp0     0.06 secs Sat Oct 21 07:01
grep            fc      ttyp0     0.08 secs Sat Oct 21 07:01
tail            fc      ttyp0     0.06 secs Sat Oct 21 07:01
```

Based on the stated assumptions, there must be a method by which modifications to the audit trails are carried out. The most common method is to *edit* a file containing the audit trail by using an on-line editor. Other options might include removing all audit entries over a time frame by using on-line tools, removing entries associated with a particular user over a particular timeframe, removing records associated with a particular program over a particular timeframe, or removing records associated with a particular file over a particular timeframe.

For example, if an attacker thought the records:

```
Oct 23 06:07:12 all sendmail[28011]: AA28011: message-id=<Pine.SUN.3.91.951023022123.9641C-100000@eskimo.com>
Oct 23 06:07:12 all sendmail[28011]: AA28011: from=<owner-cypherpunks@toad.com>, size=2075, class=-60
Oct 23 06:14:15 all sendmail[28822]: AA28011: to=<fc@all.net>, delay=00:07:05, stat=Sent
```

indicated something about an attack that the attacker wanted to cover up, they might try to remove those records from that audit trail.

One method of modification would be to use an on-line editor to edit the file containing the audit trail, remove the undesired lines, and save the resulting file. Another method for this example would be to selectively remove the lines relating to mail message *AA28011* by simply deleting all records containing *AA28011* from the audit trail.

# 3 Loss of Expected Behavior

One way to detect tampering with audit trails such as those displayed above is by looking for expected behavior that is missing. In some cases, it may be nearly impossible to remove all indicators.

In the example using *sendmail*, incomplete removal of the items corresponding to message *AA28011* might leave a single line of *AA28011* without the other lines displayed above. Since these lines always come in sequences (although not always with three entries), it is possible to detect any set of missing lines given that one line is present. Complete removal of these lines leaves us with no entries corresponding to *AA28011*, but since this is a sequence number, there will be indicators of *AA28010* and *AA28012*, leaving us with clear identification of the missing *AA28011* records. In order to tamper with this information without permitting these sorts of detection, this audit trail has to be replaced with another similar audit trail which is plausible and maintains properties consistent with the records replaced.

Sometimes tampering can be detected as a side effect of the process used to tamper. For example, the record of completed processes shown above produces several audit records per second. This is not true on all systems, but the system being examined in this instance runs a series of automated programs with great regularity. Since audit trails are often quite long (1 million bytes per day is not unusual for an active timesharing system) modifying an audit trail may take a significant amount of time. If an editor or a set of programs are used to create a forged audit trail, the overwriting of the original audit trail produces an *end-of-file* associated with the modified version of the audit trail. Since audit trails are written many times per second, this may result in gaps in the audit record corresponding to the period during which the records were being overwritten. In order to eliminate this sort of detection, audit trails must be modified in place.

Missing information makes stopping and restarting audit trails a problem as well. When audit trails are restarted, they typically indicate the restart, while the lack of audit records over a period of time is an indicator of tampering. In the same way that missing *sendmail* audit records may indicate tampering, many programs that are commonly used are used in conjunction with other programs in particular sequences and/or proportions. For example, the following sequence is repeated twice within the same minute and is likely to be repeated throughout the audit trail because it is associated with a particular automated process.

```
grep            fc       ttyp0      0.08 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
sh              fc       ttyp0      0.03 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.06 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.05 secs Sat Oct 21 07:01
...
grep            fc       ttyp0      0.05 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
sh              fc       ttyp0      0.02 secs Sat Oct 21 07:01
tail            fc       ttyp0      0.09 secs Sat Oct 21 07:01
grep            fc       ttyp0      0.06 secs Sat Oct 21 07:01
```

Another example of missing behavior is missing login or remote access information. In the following audit trail, the user connected using the *ftp* file transfer protocol twice within 21 minutes and not at all for the rest of the morning.

```
Oct 23 06:22:09 all in.ftpd[29745]: connect from unix
...
Oct 23 06:43:00 all in.ftpd[2245]: connect from unix
```

If one of these were missing and the user was sufficiently aware to be able to pick up this sort of behavior, this could be a good indicator of tampering. The problem with this sort of detection is that this sort of behavior is rarely noticed and people are rarely reliable enough to remember such minutia. The predictability of automated processes are far better indicators of tampering in most cases.

There are many other types of missing behavior that can result from attempts to tamper with audit trails.

# 4 Introduction of Unexpected Behavior

Another way that tampering may be detected is through the introduction of unexpected behavior. For example, when looking at the audit records pertaining to program usage, there were no occurrences of *root* running any process other than *sync* and *ls*:

```
sync            root     ttyp1      0.20 secs Sat Oct 21 07:01
...
sync            root     ttyp1      0.17 secs Sat Oct 21 07:01
sync            root     ttyp1      0.16 secs Sat Oct 21 07:01
ls              root     ttyp1      0.06 secs Sat Oct 21 07:01
sync            root     ttyp1      0.16 secs Sat Oct 21 07:01
sync            root     ttyp1      0.17 secs Sat Oct 21 07:01
sync            root     ttyp1      0.17 secs Sat Oct 21 07:01
sync            root     ttyp1      0.22 secs Sat Oct 21 07:01
sync            root     ttyp1      0.20 secs Sat Oct 21 07:01
```

The use of an unexpected program by a any user could be a side effect of tampering, but tampering with audit trails normally requires privileges, and privileged users usually don't run many of the same programs commonly used by unprivileged users. If we saw an entry with *root* running *vi* (the visual editor), it would be highly suspect unless an authorized user was using *vi* at or about that time. For each kind of system, there are programs that are more and less commonly used by each user. By looking for uncommon usage, we may be able to detect tampering.

# 5 Inconsistency in Audit Trails

Perhaps the most powerful method for detecting tampering with audit trails is using the correlation between different sorts of audit trails. Using the examples above, we have a sequence of events from the *syslog* audit trail:

```
Oct 23 06:03:07 all in.thttpd[27542]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
...
Oct 23 06:03:31 all in.thttpd[27572]: twist gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk unknown
Oct 23 06:03:35 all in.thttpd[27573]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
```

and a corresponding sequence of events from the *in.thttpd* audit trail:

```
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:08 cat /heaven.html
gate.uwe.ac.uk unknown 1995/10/23 06:03:32 cat /searcher/top.html
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:35 cat /admin/mlist/top.html
```

This redundancy between different audit trails is quite hard to forge. For example, the mechanisms for turning auditing on and off are different for different audit trails. If an auditable event occurs between turning off one audit trail and turning off the other, an inconsistency will result. In the case of these programs, there are at least two other audit trails generated by these events that can also be correlated to detect tampering.

Tampering without inconsistencies is harder in environments where there is more activity because there are more auditable events, which means that the windows for turning off audit trails without detection are far smaller. It may take a lot of time in the system to find all of the audit trails being generated and tamper enough to remove all of the inconsistencies. The longer the attacker is in the system, the more likely they are to make a mistake, and as they attempt to remove signs of tampering, they may generate still more audit trails, and of course they may be detected as an illicit user and traced to a source.

# 6 Automating the Analysis

To assess the difficulty associated with these methods, we wrote some simplistic analysis programs to correlate audit records.

The first program explored the problem of comparing the *twist* records from the *syslog* audit trail with records from the *in.thttpd* audit trail. For the purposes of demonstration, we used a sampling of about 40 records taken from a low-usage period on our server. Our first observation was that the sizes of the log files were different. One had 41 entries while the other had 43. This means that we should get some indication of an inconsistency because there should (in most cases) be a one-to-one correspondence between *twist* entries in the *syslog* audit trail and *cat* entries in the *in.thttpd* audit trail.

Several sources of differences such as this can be accounted for. For example, when examining an audit over a time span, there may be edge conditions where a *twist* happened

before the start of the time span in question and the corresponding *cat* happened within the time span. Similarly, a trailing *cat* can be lost at the end of the time span. Another complexity results from the limit behavior of processes in a multiprocessing environment. For example, as the number of available processes is exhausted, there may be a *twist* record corresponding to an *http* process that couldn't be started. Another cause of such a mismatch would be a case where a process that would result in a *cat* record is terminated before it can produce that record. Another example where a *twist* could be lost would be a case where different disks stored different audit trails and the disk storing *twist* records became full or had a write failure. The problem is further complicated when multiple machines are being used in the audit process. For example, many audit systems transmit audit records from one machine to another for storage. If the time stamps are different on the different machines, or if on one machine the time is changed (e.g., by the action of a systems administrator or attacker), the correlation may not be properly ordered (but the audit trails should still be in real-time sequence, leaving an indication of the change).

Before making judgments about the source of inconsistencies, it is important to examine the various ways in which they can occur and to rule out as many as possible. But without making any such judgments, it is straight forward, in this case, to find obvious inconsistencies. By understanding the way in which these programs interact, we know that for each *cat* record there should be a proceeding *twist* record with identical *user* and *system* fields (e.g., *gate.uwe.ac.uk* is a *system* and *tcprelay* is a *user*). From the example above:

```
Oct 23 06:03:07 all in.thttpd[27542]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
...
Oct 23 06:03:31 all in.thttpd[27572]: twist gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk unknown
Oct 23 06:03:35 all in.thttpd[27573]: twist tcprelay@gate.uwe.ac.uk to /usr/etc/in.thttpd gate.uwe.ac.uk tcprelay
```

For each of the entries above, there is a corresponding subsequent entry below. In this case, the times are quite close to each other. In one case both events were logged within the same second.

```
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:08 cat /heaven.html
gate.uwe.ac.uk unknown 1995/10/23 06:03:32 cat /searcher/top.html
gate.uwe.ac.uk tcprelay 1995/10/23 06:03:35 cat /admin/mlist/top.html
```

Unfortunately, there is no guarantee that the records will be this well ordered. A *cat* could take several minutes, and there could be many intervening *twist* and *cat* records from

the same user at the same machine. As a result of this analysis, these particular audit sources have now been augmented to provide process identification information that allows these audit records to be correlated very easily. A more traceable audit trail looks like this:

```
128.163.83.60 unknown 1995/10/25 12:28:34 25127 25108 cat /browse.html
128.163.83.60 unknown 1995/10/25 12:28:47 25136 25135 cat /journal/top.html
128.163.83.60 unknown 1995/10/25 12:29:18 25191 25189 cat /journal/asis/top.html
```

In this case, the *cat* record relates back to the *twist* record (i.e., *25136* is the child process of *25135* and thus correlates to the *twist* record marked with that number) and the analysis is straight forward. Unfortunately, many audit sources available with systems don't provide this sort of information. The resulting analysis is significantly more complicated and less certain as a result.

The audit trails analyzed for the purposes of this example are included in the appendices. Analysis shows that the following lines appear in the *twist* log but don't appear in the *cat* log:

```
Oct 25 12:22:12 all in.thttpd 24461 : twist cbpc38.research.aa.wl.com to
      /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:37:34 all in.thttpd 26016 : twist 128.163.83.60 to
/usr/etc/in.thttpd 128.163.83.60 unknown
```

As the last audit record in the analysis, the second entry is likely a boundary condition where the corresponding *cat* record had not yet been generated. The first of the two records is an inconsistency that likely indicates an uncompleted request in which the *http* program terminated before audit trails were generated.

An efficient algorithm for doing this analysis is easy to come up with. For example, a merge/sort of the records using the process number as the key, followed by a linear search for records that don't come in pairs yields a list of mismatches very quickly. We believe that there is a linear time algorithm for doing this analysis based on filling two tables referenced by process number as the data is read in, and then linearly searching for entries in one table that don't appear in the other table.

As a side note, process identification numbers are not unique. In most systems, they are allocated incrementally from 0 through infinity in a fixed modulus (e.g., the process after 23 is 24, and the process after 32447 is 0). In exceptionally large audit records, the analysis may be complicated by this, however, in practice, a lot of time passes between successive recurrences of the same process identification number.

# 7 Summary, Conclusions, and Further Work

We have presented some examples of how audit trails may be correlated with themselves, with each other, and with known behavior in order to detect attempts to tamper. It appears that using the techniques described here, the redundancy in audit trails can be exploited to detect inconsistencies that are indicative of tampering.

In considering this issue, complexity may become an important concern. In particular, it might be valuable to determine how complex it is to make a consistent forgery and how complex it is to detect an inconsistent forgery. Typically, an attacker has relatively little time to perform forgery, or at least the risks of detection increase with more time in the system. A defender, on the other hand, may have a lot of computer time available to look for inconsistencies. This would seem to give defenders a computational advantage, but without a detailed mathematical analysis, we cannot say this for certain.

It seems clear that by providing information that allows us to associate different audit records with each other, the complexity of correlation is dramatically reduced.

A great deal of further work may be required in order to create automated tools for detecting tampering, analyze the effectiveness of these tools, understand the implications of false positives and negatives resulting from the use of these tools, and make them a part of the overall information protection process.

It is the ultimate hope of this work to be able to resolve the guilt or innocence of accused people with a higher degree of certainty, and to strengthen our ability to differentiate reality from speculation.

# A twistlog

```
Oct 25 11:27:37 all in.thttpd 18776 : twist jazz.ml.tele.fi to /usr/etc/in.thttpd jazz.ml.tele.fi unknown
Oct 25 11:28:30 all in.thttpd 18857 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:28:36 all in.thttpd 18884 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:30:26 all in.thttpd 19079 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:30:51 all in.thttpd 19135 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:31:27 all in.thttpd 19220 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:31:47 all in.thttpd 19262 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:32:03 all in.thttpd 19306 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:32:36 all in.thttpd 19417 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:32:54 all in.thttpd 19444 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:33:35 all in.thttpd 19508 : twist ts3.teale.ca.gov to /usr/etc/in.thttpd ts3.teale.ca.gov unknown
Oct 25 11:35:40 all in.thttpd 19791 : twist unix to /usr/etc/in.thttpd unix unknown
Oct 25 11:35:45 all in.thttpd 19800 : twist unix to /usr/etc/in.thttpd unix unknown
Oct 25 11:46:45 all in.thttpd 20912 : twist 134.193.26.135 to /usr/etc/in.thttpd 134.193.26.135 unknown
Oct 25 11:46:47 all in.thttpd 20914 : twist 134.193.26.135 to /usr/etc/in.thttpd 134.193.26.135 unknown
Oct 25 11:47:13 all in.thttpd 20941 : twist 134.193.26.135 to /usr/etc/in.thttpd 134.193.26.135 unknown
Oct 25 12:10:39 all in.thttpd 23292 : twist conga.super.unam.mx to /usr/etc/in.thttpd conga.super.unam.mx unknown
Oct 25 12:10:54 all in.thttpd 23319 : twist conga.super.unam.mx to /usr/etc/in.thttpd conga.super.unam.mx unknown
Oct 25 12:11:02 all in.thttpd 23323 : twist conga.super.unam.mx to /usr/etc/in.thttpd conga.super.unam.mx unknown
Oct 25 12:11:06 all in.thttpd 23351 : twist waynepcnt.mcit.med.umich.edu to /usr/etc/in.thttpd waynepcnt.mcit.med.umich.edu unknown
Oct 25 12:11:10 all in.thttpd 23360 : twist waynepcnt.mcit.med.umich.edu to /usr/etc/in.thttpd waynepcnt.mcit.med.umich.edu unknown
Oct 25 12:11:53 all in.thttpd 23431 : twist conga.super.unam.mx to /usr/etc/in.thttpd conga.super.unam.mx unknown
Oct 25 12:12:07 all in.thttpd 23458 : twist conga.super.unam.mx to /usr/etc/in.thttpd conga.super.unam.mx unknown
Oct 25 12:15:54 all in.thttpd 23842 : twist 140.175.36.173 to /usr/etc/in.thttpd 140.175.36.173 unknown
Oct 25 12:16:05 all in.thttpd 23862 : twist 140.175.36.173 to /usr/etc/in.thttpd 140.175.36.173 unknown
Oct 25 12:20:37 all in.thttpd 24277 : twist 162.48.162.28 to /usr/etc/in.thttpd 162.48.162.28 unknown
Oct 25 12:20:57 all in.thttpd 24323 : twist cbpc38.research.aa.wl.com to /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:21:27 all in.thttpd 24384 : twist cbpc38.research.aa.wl.com to /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:22:12 all in.thttpd 24461 : twist cbpc38.research.aa.wl.com to /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:22:19 all in.thttpd 24462 : twist 162.48.162.28 to /usr/etc/in.thttpd 162.48.162.28 unknown
Oct 25 12:22:57 all in.thttpd 24540 : twist cbpc38.research.aa.wl.com to /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:23:19 all in.thttpd 24570 : twist cbpc38.research.aa.wl.com to /usr/etc/in.thttpd cbpc38.research.aa.wl.com unknown
Oct 25 12:25:37 all in.thttpd 24826 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:25:43 all in.thttpd 24828 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:28:34 all in.thttpd 25108 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:28:46 all in.thttpd 25135 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:29:18 all in.thttpd 25189 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:29:47 all in.thttpd 25242 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:29:55 all in.thttpd 25269 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:33:30 all in.thttpd 25627 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:35:40 all in.thttpd 25834 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:37:24 all in.thttpd 26014 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
Oct 25 12:37:34 all in.thttpd 26016 : twist 128.163.83.60 to /usr/etc/in.thttpd 128.163.83.60 unknown
```

# B httplog

```
jazz.ml.tele.fi unknown 1995/10/25 11:27:38  18778 18776 cat /browse.html
ts3.teale.ca.gov unknown 1995/10/25 11:28:30  18883 18857 Error:Can't stat file   GET /new.html
ts3.teale.ca.gov unknown 1995/10/25 11:28:36  18885 18884 cat /index.html
ts3.teale.ca.gov unknown 1995/10/25 11:30:26  19083 19079 cat /hudsonoh/top.html
ts3.teale.ca.gov unknown 1995/10/25 11:30:51  19136 19135 cat /heaven.html
ts3.teale.ca.gov unknown 1995/10/25 11:31:27  19236 19220 cat /index.html
ts3.teale.ca.gov unknown 1995/10/25 11:31:47  19263 19262 cat /browse.html
ts3.teale.ca.gov unknown 1995/10/25 11:32:04  19307 19306 cat /journal/top.html
ts3.teale.ca.gov unknown 1995/10/25 11:32:36  19418 19417 cat /journal/netsec/top.html
ts3.teale.ca.gov unknown 1995/10/25 11:32:54  19445 19444 cat /index.html
ts3.teale.ca.gov unknown 1995/10/25 11:33:41  19543 19508 cat /journal/netsec/9511.html
unix unknown 1995/10/25 11:35:41  19799 19791 cat /index.html
unix unknown 1995/10/25 11:35:45  19801 19800 cat /browse.html
134.193.26.135 unknown 1995/10/25 11:46:45  20913 20912 cat /index.html
134.193.26.135 unknown 1995/10/25 11:46:47  20915 20914 cat /allnet.gif
134.193.26.135 unknown 1995/10/25 11:47:13  20942 20941 cat /index.html
conga.super.unam.mx unknown 1995/10/25 12:10:40  23315 23292 cat /index.html
conga.super.unam.mx unknown 1995/10/25 12:10:54  23338 23319 cat /allnet.gif
conga.super.unam.mx unknown 1995/10/25 12:11:02  23350 23323 cat /integ/top.html
waynepcnt.mcit.med.umich.edu unknown 1995/10/25 12:11:06  23352 23351 Error:Unknown request   HEAD /journal/top.html
waynepcnt.mcit.med.umich.edu unknown 1995/10/25 12:11:11  23372 23360 cat /journal/top.html
conga.super.unam.mx unknown 1995/10/25 12:11:53  23432 23431 cat /heaven.html
conga.super.unam.mx unknown 1995/10/25 12:12:08  23459 23458 cat /help.html
140.175.36.173 unknown 1995/10/25 12:15:54  23843 23842 cat /index.html
140.175.36.173 unknown 1995/10/25 12:16:05  23870 23862 cat /allnet.gif
162.48.162.28 unknown 1995/10/25 12:20:37  24310 24277 cat /index.html
cbpc38.research.aa.wl.com unknown 1995/10/25 12:20:57  24355 24323 cat /allnet.gif
cbpc38.research.aa.wl.com unknown 1995/10/25 12:21:27  24410 24384 cat /heaven.html
162.48.162.28 unknown 1995/10/25 12:22:19  24489 24462 cat /browse.html
cbpc38.research.aa.wl.com unknown 1995/10/25 12:22:57  24566 24540 cat /tests/top.html
cbpc38.research.aa.wl.com unknown 1995/10/25 12:23:19  24596 24570 cat /tests/one time test.html
128.163.83.60 unknown 1995/10/25 12:25:37  24827 24826 cat /index.html
128.163.83.60 unknown 1995/10/25 12:25:44  24829 24828 cat /allnet.gif
128.163.83.60 unknown 1995/10/25 12:28:34  25127 25108 cat /browse.html
128.163.83.60 unknown 1995/10/25 12:28:47  25136 25135 cat /journal/top.html
128.163.83.60 unknown 1995/10/25 12:29:18  25191 25189 cat /journal/asis/top.html
128.163.83.60 unknown 1995/10/25 12:29:47  25243 25242 cat /journal/netsec/top.html
128.163.83.60 unknown 1995/10/25 12:30:01  25270 25269 cat /journal/netsec/9511.html
128.163.83.60 unknown 1995/10/25 12:33:32  25628 25627 cat /journal/netsec/9510.html
128.163.83.60 unknown 1995/10/25 12:35:42  25835 25834 cat /journal/netsec/9509.html
128.163.83.60 unknown 1995/10/25 12:37:24  26015 26014 cat /journal/csi/top.html
```