The Design of an Audit Trail Analysis Tool

Eric A. Fisch, Gregory B. White, and Udo W. Pooch[†]

Department of Computer Sciences
Texas A&M University
College Station, Texas 77843

Abstract

This paper discusses the design of a tool that automatically removes security-sensitive information from intruder activity log files collected at a compromised site. The sanitization of sensitive information will enable researchers to further study the log files without further compromising the security of the affected sites. The paper begins with a brief discussion of the importance of such a tool and a description of the complete sanitization process. This follows with an examination of the important design issues of the sanitizer. The paper concludes with the final design of a sanitizer for SunOS based intruder activity logs.

1 Introduction

The number of users and computer systems connected to large networks is continuously increasing [4,5]. Unfortunately, the number of reports of attempted intruder break-ins is also increasing [2]. Numerous methods have been devised to keep intruders out, but all too often a new security hole is found or a new method developed to circumvent existing security controls. In response to this, many sites have begun to log every user's actions [1]. When properly analyzed, these activity logs provide useful information resulting in bug reports and system enhancements [1]. Additionally, the logs may assist in the characterization of intruders and intrusive behavior [6,7]. Using the information contained within the logs, researchers may properly define an intrusion and determine when one has occurred. Typical intrusive behavior becomes more discernible when multiple intrusion logs are cross-referenced and compared [7]. This can result in the creation of an intruder profile- something many researchers believe to be useful in detecting an intruder's presence [2,6,7]. Future damage assessment also benefits from the knowledge of typical intruder actions. Many of these benefits, however, are reduced because log files tend to contain sensitive data that details security weaknesses and private or proprietary information that cannot be released to the public. On the rare occasion that incident details are released they are vague synopses of the incident that provide limited value.

In an effort to release information and coordinate the development of countermeasures against intruders, many of these logs have been turned over to the Computer Emergency Response Team (CERT). CERT performs their own analysis and issues notices and warnings as they deem necessary. Logs are delivered to CERT under the condition that they not be redistributed. Distribution of these logs, however, may assist other research efforts. Wide distribution of the logs therefore requires that they are sanitized. Sanitization is the process of removing, or concealing, sensitive information from a file without effecting the presentation of the concepts within the file. Currently the only method available to perform a sanitization is through an exhaustive, manual editing session. The logs, however, are too large and numerous for this to be viable [7]. To rectify this problem a tool to sanitize the data was designed and is discussed below.

This paper begins with a discussion of the intruder data collection process. This is followed with a discussion of the sanitization process. The various types of sanitization are then presented. This paper continues with a discussion of automation and sanitization and concludes with the design of the sanitizer and its template extension system.

2 Background

Based on incidents studied at Texas A&M University, the information within log files tends to be data files from compromised accounts, audit trails and log files, and network traffic that has been recorded during the intrusion [3]. Data files consist of miscellaneous information that does not contribute to the analysis of a break-in. Audit trails and log files provide more information than data files, but usually contain small amounts of data from which to determine a hacker's activities. Recorded network traffic, however, is the best source of information since all portions of a hacker's

† This research was supported in part by Trident Data Systems and the Air Force Cryptologic Support Center; contract number 32525-43750.

communication channel are recorded and the actions reproducible.

While network traffic provides the greatest amount of useful information, all data must follow a four step process to provide a useful analysis. Figure 1, depicts a model of the process as applied to sites that collect break-in data. This model consists of four different sets of tools: loggers, converters, sanitizers, and analysis tools. The first set of tools, loggers, take real time network traffic and record it to non-volatile storage media. This normally takes place at the site experiencing the break-in and uses some type of network listening, or sniffer, device. Examples of such currently available tools are *etherfind*, *snoop*, and *tcpdump*.

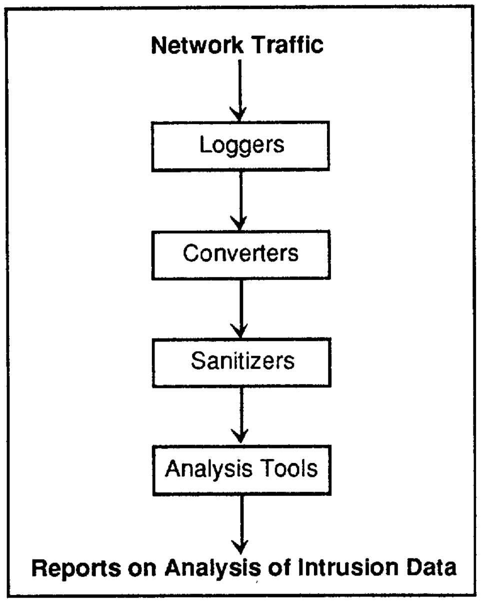

Figure 1. Model of Intrusion Data Collection Process.

The second set of tools are converters. Converters take the output from loggers and convert it to typed data. Typed data is a high level protocol stream such as a telnet session, series of NFS transactions, or interactive shell session. The typed data used in this paper are interactive shell sessions. This was selected as it is commonly found in intruder logs and is a large portion of the Texas A&M University incident logs. Once the input is converted, the typed data gets piped into a sanitizer.

The third type of tools, sanitizers, take the typed data from the converters and produce a data set with all sensitive information removed. Sanitization is done to prevent future attacks at any of the sites involved and still provide useful information for others to analyze. The sanitizers are ideally used at the site experiencing the break-in. This allows the site to sanitize proprietary information as well as security-sensitive information. While organizations such as CERT may use these tools, they will not be able to effectively determine what information is proprietary and should be sanitized.

The fourth set of tools are analysis tools. Along with user assistance, they perform a majority of the data interpretation. Development of tools to perform an analysis must wait until the sanitizers are developed and produce information to analyze. They are therefore not addressed at this time.

Once the log files have passed through the four sets of tools, detailed information about an intruders actions can be gathered. This information can then be used in conjunction with other log files to generate profiles and improve site security. Before the analysis can occur, however, sanitizers and analyzers must be developed. The first step in this process, development of a sanitizer, is discussed below.

3 Sanitization

The concept of automatic sanitization is simple; the computer must remove any information that can identify any specific site or user account involved in an attack. To sanitize the information and protect attacked sites from future aggressions, every occurrence of account names, user names, site names and IP addresses must be altered or removed. These security-sensitive items will appear in many locations throughout the data. An analysis of the intrusion files gathered at Texas A&M University led to the list of items that contain sensitive data listed in Figure 2 [3]. This list is based on the use of SunOS 4.1 at the compromised site. The list will vary slightly with the use of a different operating system and may therefore be incomplete. Such a list, while possibly incomplete, will assist in development of the sanitizer.

- System prompts
- System files such as /etc/passwd and /etc/group
- Command arguments and output
- Mail files
- Visible user files
- Hidden user files such as .netrc
- Source code
- File and directory names
- Messages of the day or login messages
- System banners

Figure 2. Items Containing Sensitive Data.

The last item listed in Figure 2, system banners, may not necessarily state a site name but must still be examined and possibly sanitized. Many systems welcome users with a banner or logo that contains a textual picture which identifies the site. For example, a series of 'X' characters that take the shape of a company or school logo. Since these banners also identify the site, they must be sanitized.

3.1 Methods of Sanitation

Once a site determines what information requires sanitization, the degree of sanitization applied must be decided. The degree of sanitization may vary from retaining only the most basic information to retaining as much data as possible. This will depend upon a user's desire to assist the process and the usefulness of the output logs. The following four methods of sanitization are therefore presented: (1) simple sanitization, (2) information-tracking sanitization, (3) format sanitization, and (4) comprehensive sanitization. Each of these methods requires varying degrees of user assistance and produces varying degrees of output log usefulness.

The first method of sanitization, simple sanitization, is the most basic. Simple sanitization entails the creation of a document listing the specific commands issued by the intruder. Figure 3 contains an example of the output for this level of sanitization. As is shown, neither input arguments nor the results of these commands are saved in the output file. Sanitizing at this level has the benefit of being easily automated. Large amounts of information, however, will be lost. For example, the output might indicate that mail was sent or files were edited, but it will not indicate where mail was sent to nor will it indicate which files were modified or copied. Knowing that an intruder sent six mail messages is not as useful as knowing that all six messages were sent to the same individual. Knowing that an intruder edited a file as opposed to knowing that the intruder edited a file named 'pswdcrack.c'. It becomes nearly impossible to determine whether activities are as harmless as fixing the spelling in a document or as dangerous as editing the password files. This method of sanitization might be useful in developing intruder signatures but it removes too much useful information.

```
rusers
finger
finger
ls
who
cat
ftp
vi
cc
FILE
```

Figure 3. Simple Sanitization.

The second method of sanitization, information-tracking sanitization, adds to the usefulness of the output by tracking sensitive information through the log files. The sanitizer keeps a symbol table of sensitive information that must be removed or replaced. When information not already in the table is found and must be sanitized, it is assigned a unique marker and entered in the table. When previously identified data needs to be sanitized, the table is examined and the data replaced by the associated identifier. For example, all references to a single user or file are retained and noted in the output. Figure 4 contains a sample of a fictitious information-tracking output file.

```
SYSTEM1% rusers
SYSTEM1% finger USER1
SYSTEM1% finger USER2
SYSTEM1% ls ~USER1
SYSTEM1% who
SYSTEM1% cat FILE1
SYSTEM1% ftp SYSTEM2.DOMAIN1
ftp> get FILE2
ftp> bye
SYSTEM1% vi FILE2
SYSTEM1% cc FILE2
SYSTEM1% FILE3
```

Figure 4. Information-Tracking Sanitization.

Information-tracking sanitization can be extended to include the output of certain system commands such as `who`, `finger`, or `rusers`. Commands such as these have consistent output formats that make their sanitization simple. Figure 5 provides an example of the sanitized output from the SunOS 4.0 `finger` command. The specific commands whose output needs sanitization, however, is operating system dependent.

The third method of sanitization is format sanitization. Format sanitization goes a step beyond the information-tracking method of sanitization by suppressing as little information as possible. The advantage of this method is the handling of different types of data formats. Instead of just examining a text file the sanitizer will locate segments of information that have been re-formatted. An example of this would be a uuencoded file. Other recognized formats include tar, compression and gnu-zip. While this process provides

```
SYSTEM2% finger USER3
Login name: USER3                    In real life: INDIVIDUAL4
Directory: /user/USER3               Shell: /bin/csh
On since Feb 25 09:01:24 on ttyp3 from SYSTEM4:0.0
21 seconds Idle Time
No unread mail
No Plan.
SYSTEM2%
```

Figure 5. Information-Tracking Sanitization of a system command.

useful information, it proves to be quite difficult for two reasons. First, it is difficult to create the necessary patterns that indicate the beginnings and endings of the data. Second, a segment may have to go through reformatting numerous times before it is a recognizable textual format that the sanitizer can process. For example, a segment could have been tar-ed and compressed not just once, but numerous times. An organization must therefore decide how many times a sanitizer will process any given segment before the segment is deleted.

The fourth method of sanitization, comprehensive sanitization, builds upon the lower methods by examining every segment of the log file. Comprehensive sanitization includes the examination of source code. Accordingly, methods for properly sanitizing popular programming

```
SYSTEM6% cat FILE1
From USER16@SYSTEM14.DOMAIN3 Mon Dec  6 12:42:52 1993
Received: from DOMAIN4 by SYSTEM7 (AA25175); Mon, 6 Dec 93 12:42:48 CST
Received: from SYSTEM15.DOMAIN3 by DOMAIN4 with SMTP id AA11685
  (5.65c/IDA-1.4.4 for <@DOMAIN4:USER1@SYSTEM1.DOMAIN5>); Mon, 6 Dec 1993
11:43:33 -0700
Received: from SYSTEM14.DOMAIN3 by SYSTEM15.DOMAIN3 via SMTP
(920330.SGI/920502.SGI.AUTO)
     for @DOMAIN4:USER1@SYSTEM1.DOMAIN5 id AA20005; Mon, 6 Dec 93
11:35:04 -0700
Received: by SYSTEM14.DOMAIN3 (930416.SGI/920502.SGI)
     for @SYSTEM15.DOMAIN3:USER!@SYSTEM1.DOMAIN5 id AA18445; Mon, 6 Dec
93 11:33:49 -0700
Date: Mon, 6 Dec 93 11:33:49 -0700
From: USER16@SYSTEM14.DOMAIN3 (INDIVIDUAL1)
Message-Id: <9312061833.AA18445@SYSTEM14.DOMAIN3>
To: USER1@SYSTEM1.DOMAIN5
Subject: Hello
Status: R


INDIVIDUAL2,

  Just a quick note to say hi and give you my new email address.  I'm
now at USER16@DOMAIN3.  I do not have a new work phone # yet.  When I
get one, I'll let you know.

-INDIVIDUAL4
```

Figure 6. An Example of Comprehensive Sanitization.

languages should be included. It is also in this method that the sanitization of text files, such as email, occurs. Both email headers and the body of the email must be sanitized. The sanitization must examine and replace system names, domain names and user names. Figure 6 contains an example email message that was sanitized.

In addition to source code and email, comprehensive sanitization provides the user the capability to sanitize information of a proprietary nature. For example, proprietary information about an organization's new product line should be sanitized. While previously discussed sanitization methods will completely remove proprietary data, it should be handled separately to reduce information loss. To accomplish this, the sanitizer will build a dictionary of commonly used words and phrases that will assist in identifying and sanitizing previously unrecognized information. With each use of the sanitizer, an organization's dictionary will identify an increasing amount of the proprietary information. Thus, the use of a dictionary at the comprehensive sanitization level enables the user to customize the sanitizer to suit their needs.

The four sanitization methods provide different amounts of sanitization. With each successive method the usefulness of the output increases. To obtain the additional usefulness provided by each method, a site must provide additional user assistance. To encourage the use of the sanitize, it is important to automate the process as much as possible.

3.2 Automatic Sanitization

One of the benefits of using an automated sanitizer is that much of the tedious editing is automatic. The process, however, is not fully user assistance-free. The amount of user assistance required is directly related to the amount of useful information that will be gained from the process. This is made evident by looking at the human intervention required at each level of sanitization. Simple sanitization produces output of little value and requires little human assistance. Comprehensive sanitization, however, produces useful output but requires greater user assistance. An examination of the comprehensive sanitization process reveals why more user assistance is necessary.

The nature of comprehensive sanitization makes it difficult to fully automate. Assistance will be necessary for the tool to build up a list of user-defined sensitive words. This requires the user to respond to system prompts when the sanitizer locates unfamiliar data. Thought will be required for making these decisions, as not every occurrence of a given word will require sanitization. It may also be the case where every occurrence of a word should not be sanitized. For example, when the user name 'white' appears, the sanitizer should replace it with an identifier token that may be used for future references to that user. In the event that a sentence like "The United States flag has red and white stripes" follows, however, the word 'white' should not be sanitized. Were this to happen, 'white' would be easily inferred as the user name. It becomes necessary to provide some method of determining which words should remain unchanged and which should be sanitized. While a natural language parser could accomplish this task, in the near term a user will oversee the process. Fortunately, sanitization gradually shifts from being a heavily user assisted process to requiring only occasional assistance. The sanitization process will always require some degree of direction and assistance, however, as the sanitizer will frequently encounter new information. Nonetheless, the sanitizer should require little user assistance from the onset.

4 Sanitizer Package Design

The sanitizer package is designed to provide a flexible and automated means of sanitization. To achieve this, the sanitizer package consists of a limited functionality sanitizer and a set of sanitizer rules, called templates. By itself, the sanitizer provides only simple level sanitization. When various template sets are applied to the sanitization process, they enable the sanitizer to function at the information tracking, format, and comprehensive sanitization levels. This sanitizer package design provides both flexibility and automation. The incorporation of these characteristics in the design of the sanitizer and template system is discussed below.

```
Pattern:     finger
Type:        command
Format:      finger <username>
Response:    <<begin multiple>>
             Login name: <username>  In real life: <individual>
             Directory: <directory>  Shell: /bin/csh
             <<delete>>
             <<end multiple>>
```

Figure 7. An Example Template.

4.1 Sanitizer Design

The basic design of the sanitizer tool is similar to that of a spell checker. The sanitizer scans the input data file and attempts to apply any of the user selected rules. When information is not sanitized by existing rules, a user may create a new rule that sanitizes the information. The newly created rule may also be added to the rule base and used in future sanitizations. Eventually, an organization will have a set of rules that properly sanitizes a data file with little user assistance. The sanitizers ability to function in an automated manner and learn from a user results in an intelligent and flexible tool.

To further promote flexibility and make the sanitizer package easy to use, the sanitizer will provide up to date information concerning the sanitization process. To accomplish this, the sanitizer will have a graphical user interface that displays the data being sanitized and the rules used in the sanitization process. A user should also be able to adjust the sanitization process as the data and their sanitization needs dictate. Therefore, the sanitizer will provide a user the means to select the rules used by the sanitizer, to save and recall sets of rules, and to create and apply new rules throughout the sanitization process. This interactive design enables an organization to easily modify the sanitization process to suit their needs.

4.2 Template Design

An integral part of the sanitizer package is the set of templates used to determine what the system will sanitize. Each template consists of four parts: a pattern, a pattern type, a command and parameter format, and the system response. The pattern field indicates the data pattern that the sanitizer locates. The type field of the template specifies what kind of pattern is matched. The word 'finger', for example, may appear as prose in the body of a document as opposed to a command. The type field prevents the sanitizer from pattern matching in this situation. The format field indicates the context in which the pattern is found. This may include the specific item, or argument, to be sanitized. The response field indicates how the expected output should be sanitized. An example of the template for the SunOS 4.0 `finger` command is provided in Figure 7. The figure also demonstrates the use of template indicators. Indicators identify the items that require sanitization and sanitizer commands.

Indicators are scattered throughout the template to assist the sanitizer in properly sanitizing the input logs. Indicators are denoted by the bracket symbols ('<' and '>') surrounding them. Item indicators, enclosed in single brackets, are the items that will be replaced by the sanitizer. When the input data is sanitized, all items that correspond to one of the single bracketed indicators are replaced with markers. A list of markers and what they identify is recorded so that the same item does not correspond to multiple markers. The indicator inside of the single brackets tells the sanitizer what type of marker should replace the original information. For example, every occurrence of a specific username in the finger command would be replaced with a username token such as USER7 in the final output. In addition to item indicators, brackets are used to identify commands to the sanitizer.

Sanitizer commands are surrounded by two pairs of brackets (i.e. '<<' and '>>'). These brackets indicate commands to the sanitizer describing special actions the sanitizer should take. Figure 7 contains three special commands. The first and third commands indicates that multiple occurrences of the same information are possible. The second special command indicates that everything following the `Directory:` line should be deleted. Additional sanitizer commands indicate how information in the data should be parsed for sanitization, what actions the sanitizer should take when sanitizing, and whether user a decision is required from the user.

To relieve a user from learning all of the special commands used by the sanitizer, a stand alone template editor will be included in the sanitizer package. The template editor will assist in creating and modifying templates. The editor will provide the user with numerous skeletal statements that follow the required template format. A user need only fill in the blanks of a statement to make a template function properly. While any text editor may modify or create templates, the template editor will relieve a user from unnecessarily learning about template semantics and syntax.

5 Summary

Organizations no longer have the resources to devote personnel to the manual sanitization of intruder attack logs. In response to the these problems, an automatic sanitization tool was designed. This tool will allow organizations to sanitize large amounts of data with little user assistance. This enables others to safely share attack information and thus develop better defense mechanisms. The sanitizer is designed to be easily tailored to suit an organization's needs. The sanitizing tool allows the organization to select the type of sanitization that should be applied to a log file. Furthermore, a database of organization-dependent items that are sanitized, such as proprietary information, is kept for future sanitizations. The flexibility and power of this sanitization tool makes it ideal for producing information that can be used in the fight against intruders.

Acknowledgements

The authors wish to thank Paul Brutch, David Hess, and Douglas Schales for their assistance.

References

[1] Kaplan, Ray and Clyde, Robert, Learning from the Loss: Classic VMS Security Breaches, *InfoSecurity News*, May/June 1993, p. 28-29.

[2] Lunt, Theresa F., Automated Audit Trail Analysis and Intrusion Detection: A Survey, *Proceedings of the 11th National Computer Security Conference*, October 1988, p. 65-73.

[3] Safford, David R., Schales, Douglas Lee and Hess, David K., The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment, *Proceedings of the fourth USENIX Security Symposium*, 1993.

[4] Sanberg, Jared, Security Breach at the Internet Raises Worries, *The Wall Street Journal*, February 7, 1994, p. B8.

[5] Ubois, Jeff, The Internet Today, *SunWorld*, April, 1993, p. 90-95.

[6] van Horne, J. and Halme, L., *Analysis of Computer System Audit Trails - Initial Data Analysis*, Sytek Technical Report TR-85009, Mountain View, California, September 1985.

[7] Weiss, Winfried R. E., and Baur, Adalbert, Analysis of Audit and Protocol Data Using Methods from Artificial Intelligence, *Proceedings of the 13th National Computer Security Conference*, October 1990, p. 109-114.