Utah State University
DigitalCommons@USU

All Graduate Theses and Dissertations | Graduate Studies, School of

12-1-2008

A Novel Authentication And Validation Mechanism For Analyzing Syslogs Forensically

Steena D.S. Monteiro
Utah State University

Recommended Citation
Monteiro, Steena D.S., "A Novel Authentication And Validation Mechanism For Analyzing Syslogs Forensically" (2008). *All Graduate Theses and Dissertations.* Paper 198.
http://digitalcommons.usu.edu/etd/198

This Thesis is brought to you for free and open access by the Graduate Studies, School of at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact becky.thoms@usu.edu.

A NOVEL AUTHENTICATION AND VALIDATION MECHANISM FOR ANALYZING SYSLOGS FORENSICALLY

by

Steena D. S. Monteiro

A thesis submitted in partial fulfillment of the requirement for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Robert F. Erbacher	Dr. Chad Mano
Major Professor	Committee Member
Dr. Dan Watson	Dr. Byron Burnham
Committee Member	Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah
2008

Copyright © Steena D. S. Monteiro 2008

All Rights Reserved

ABSTRACT

A Novel Authentication and Validation Mechanism for

Analyzing Syslogs Forensically

by

Steena D. S. Monteiro, Master of Science

Utah State University, 2008

Major Professor: Dr. Robert F. Erbacher
Department: Computer Science

This research proposes a novel technique for authenticating and validating syslogs for forensic analysis. This technique uses a modification of the Needham Schroeder protocol, which uses nonces (numbers used only once) and public keys. Syslogs, which were developed from an event-logging perspective and not from an evidence-sustaining one, are system treasure maps that chart out and pinpoint attacks and attack attempts. Over the past few years, research on securing syslogs has yielded enhanced syslog protocols that focus on tamper prevention and detection. However, many of these protocols, though efficient from a security perspective, are inadequate when forensics comes into play. From a legal perspective, any kind of evidence found at a crime scene needs to be validated. In addition, any digital forensic evidence when presented in court needs to be admissible, authentic, believable, and reliable. Currently, a patchy log on the server side and client side cannot be considered as formal authentication of a wrongdoer. This work presents a method that ties together, authenticates, and validates all the entities involved in the crime scene—the user using the application, the system that is being used,

and the application being used on the system by the user. This means that instead of merely transmitting the header and the message, which is the standard syslog protocol format, the syslog entry along with the user fingerprint, application fingerprint, and system fingerprint are transmitted to the logging server. The assignment of digital fingerprints and the addition of a challenge response mechanism to the underlying syslogging mechanism aim to validate generated syslogs forensically.

(61 pages)

For my parents who have always supported and encouraged my love for science and learning through the years.

ACKNOWLEDGMENTS

A great deal of thanks goes out to Dr. Robert F. Erbacher for his immense support, patience, knowledge, and guidance. My work is most certainly a reflection of these. His unfailing confidence in me has always been an encouraging factor during certain challenging phases of my research. I would like to thank Dr. Chad Mano for encouraging me during my first semester to do a master's thesis and for introducing me to the wonderful world of wireless security through his class. Also, I would like to thank Dr. Dan Watson for being on my committee.

A thank you to all my lab mates (and friends), and my special friends in the Computer Science Department for their unwavering support, help, and jolly company without which my work in the lab, at my research, and at school would have been mundane and dull. It is always a pleasure to work with you and discuss the "finer aspects" of security, forensic evidence, and random geek humor.

My family, though thousands of miles away, has been my emotional backbone through my time here at Utah State University. I would like to acknowledge and thank parents for their love, sacrifices, and support through the past years. They have made me what I am today. Also, I'd like to express my gratitude to my small (I like this word!) sister Simonah, who firmly maintains that her sister can excel at anything; wherever and whenever.

Lastly, and not in any means a small measure, I would like to thank Infant Jesus for guiding, guarding, and blessing me.

Steena D.S. Monteiro

CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS vi

LIST OF TABLES x

LIST OF FIGURES xi

CHAPTER

1. INTRODUCTION 1

1.1 Background 1
1.2 Syslog BSD 1

1.2.1 Simplicity 2
1.2.2 Flexibility 2

1.3 Syslog Security Research 3
1.4 Syslog Tools 3
1.5 Importance of this Research 3

2. SYSLOGS 6

2.1 Syslogs Thus Far 6
2.2 Background of the Proposed Method 7
2.3 Previous Syslog Research 8
2.4 Weaknesses of the Syslog Protocol 11

2.4.1. Compromising the Authenticity 11
2.4.2. Compromising the Confidentiality 12
2.4.3. Compromising the Integrity 12

2.5 Forensic Requirements 13

2.5.1 Preservation 14
2.5.2 Identification 14
2.5.3 Extraction 14
2.5.4 Documentation 15
2.5.5 Interpretation of Data 15
2.5.6 Confidentiality 15

2.5.7 Integrity 16
2.5.8 Authenticity 17

2.6 Syslog Variants 17

2.6.1 Syslog-Sign 17
2.6.2 Syslog-Auth 18

3. THE PROPOSED METHOD 19

3.1 Overview of the Proposed Method 19
3.2 The Proposed Method 21

3.2.1 Phase 1: User Authentication 21
3.2.2 Phase 2: System Connection Establishment 22
3.2.3 Phase 3: System Connection Establishment Response 22
3.2.4 Phase 4: Application Event Entry Generation 23
3.2.5 Phase 5: Application Termination 23
3.2.6 Phase 6: System Connection Termination 23

4. FINGERPRINTS AND AUTHENTICATION TRACES 24

4.1 User Fingerprints 24
4.2 Application Fingerprints 25
4.3 System Fingerprints 26
4.4 Fingerprint Generation 27

4.4.1 User Fingerprint Generation 27
4.4.2 System Fingerprint Generation 28
4.4.3 Application Fingerprint Generation 29

4.5 Authentication Traces 30
4.6 Nonces 30

5. ATTACK BACKTRACKING 32

5.1 Backtracking to an Attack 32
5.2 Reconstructing Fingerprints 34

6. FORENSIC VIABILITY 36

6.1 Requirements of Forensic Evidence 36
6.2 Evidence Certainty Levels 36
6.3 Authentication Traces and Syslogs under Certain Scenarios 38

6.3.1 Scenario One: Syslog File Deletion 38

6.3.2 Scenario Two: Spurious Entry Injection into the Syslog File.....39
6.3.3 Scenario Three: Application Updates 39

7. PROTOCOL RESILENCE 41

7.1 Attacks Against the Challenge Response System.....41

7.1.1 Phase 2: System Connection Establishment 41
7.1.2 Phase 3: System Connection Establishment Response.....41
7.1.3 Phase 4: Application Event Entry Generation 42
7.1.4 Phase 5: Applications Termination.....42
7.1.5 Phase 6: System Connection Termination 42

7.2 Attacks Against the Syslog File 42
7.3 Denial of Service.....43
7.4 Abusing Privileges 43
7.5 Application Updates.....43

8. CONCLUSION.....45

8.1 Current Scenarios.....45
8.2 Future Work 46

REFERENCES 48

LIST OF TABLES

Table Page

6.2 Mapping the certainty levels defined in [2] to syslog files.....37

LIST OF FIGURES

Figures Page

3-1 View one of overview of the proposed method 19

3-2 View two of overview of the proposed method 21

4-1 User fingerprint generation 28

4-2 System fingerprint generation 30

4-3 Application fingerprint generation 31

CHAPTER 1

INTRODUCTION

1.1 Background

Computer evidence has become an indispensible factor in proving criminal and civil cases in a court of law. Previously, the lack and/or misinterpretation of computer evidence have been the primary causes for stalling the pursuit of computer crime cases in court. Due to the lack of a formal forensic procedure, the interpretation of computer evidence at a crime scene has been often best left to expert witness testimonies. The federal rules of evidence now consider and treat computer evidence as they would documentary evidence. This means that like documentary evidence, computer evidence needs to be verified and authenticated. It is now mandatory for forensic computer expert witnesses to be able to concretely verify and defend their observations and inferences with regard to the processes followed and the tools used at the crime scene. Therefore, it is extremely important that computer evidence processing be done correctly in criminal cases. A crucial aspect of computer evidence is the documentation associated with it [1].

1.2 Syslog BSD

Syslogs, as defined by the syslog Berkeley software distribution (BSD) protocol, were developed as event reporting systems. This protocol aimed to serve as an indication that events with certain priorities occurred over time on a network. The protocol assumes every entity to be independent of each other and does not provide for any kind of binding mechanism in place when events are associated with multiple

entities on the network. The syslog protocol was devised with the following fundamental tenets in mind.

1.2.1 Simplicity

Syslogs rely on the integrity of the underlying system that implements it for its security. The original BSD protocol does not have any security considerations in place for securing or protecting the messages that are transferred to and from the systems on the network and the central logging repository. Therefore, after the transmission of a syslog message, no explicit notification is sent out to the system that generates it.

With protocol simplicity as the basic focus, UDP is the designated protocol used [2] to transmit syslogs generated by the systems on the network to a centralized logging system. The simple format of a syslog entry and the vulnerable protocol used to transmit it makes syslogs very unsubstantial evidence and causes the integrity of the entries can be challenged [3].

1.2.2 Flexibility

Syslogs are flexible in the sense that an administrator on a network can configure logging settings and logging paths. This means that the .conf file can be modified in order to change logging settings on the network. Additionally, the administrator can determine the message logging procedure. For example, messages of higher priority get logged on a different server, messages signaling the launch of applications to another, etc.

Thus, having been developed as electronic signaling systems for a network, syslogs do not have any mechanism in place that meets the security goals of authenticity,

confidentiality, and integrity. With this as the prime focus, security research has focused on cementing the existing syslog with various new techniques and methods.

1.3 Syslog Security Research

Security research has worked on enhancing syslogs in two ways: remodeling the existing syslog protocol and suggesting various architectures, which add a level of security to log files as a whole. This is discussed in Chapter 2, which looks at the background of syslogs and their security.

1.4 Syslog Tools

In order for any kind of evidence to be used in a court of law, it has to be deemed as admissible, authentic, believable, and reliable. Currently, there are several commercial tools available in the market that can be used to analyze the humongous amount of information that syslogs contain. These tools basically classify the information that a log file contains. Another trait common to all these tools is that none of them assigns any kind of forensic credibility to the entities involved in the generation of log entries. Further, most tools concentrate all attention on the final and central log file, and the collation and categorization of the information that it contains [4].

1.5 Importance of This Research

- Electronic evidence in a court of law is now treated in the same way as documentary evidence. This means that similar to its real-world counterpart, electronic evidence needs to satisfy similar phases of evidence authentication and validation. Thus, electronic evidence has to be made admissible, authentic, believable, and reliable.

- Forensic analysis can never be carried out on the actual artifact. The analysis is required to be carried out on *copies* of the artifact. The method proposed in this research satisfies this requirement, thereby satisfying a crucial forensic requirement.
- The mechanism proposed in this research has successfully assigned an identity, i.e., a fingerprint, to every entity involved in the generation of a syslog entry, thereby overwriting the principle of the original syslog BSD that emphasizes the independence of every entity.
- An attacker who attacks a network and causes havoc will firstly seek to eliminate any kind of evidence that indicates the presence of the attack or the attacker. In the event of either of these occurring, an alternate mechanism is needed whereby the system can be brought back up again and recovered to a stable state. The technique presented in this research accomplishes this.
- An important aspect to be noted is that the proposed method clearly satisfies the following authentication requirements of electronic evidence documented by the Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice [5]:
    1. Verification of the authenticity and prevention of the alteration of computer records. This is achieved through the use of the authentication traces and the challenge response mechanism.
    2. Establishment of the reliability of computer programs. This is achieved through the use of application fingerprints.

3. Identification of the author of computer-stored records. This is achieved through the use of user fingerprints.

CHAPTER 2

SYSLOGS

2.1 Syslogs Thus Far

This research provides a mechanism that validates and authenticates syslogs for computer forensic analysis. Syslogs are often smoking guns [6] in an organization wherein a computer or network attack has occurred due to the immense amount of information contained therein. Syslogs may also contain evidence of illegal or inappropriate activity by the user of an individual system. Traditionally, when computer evidence needs to be collected, the entire system is taken off-line, and the entire hard drive treated as evidence. With a network attack, there could be evidence in syslog files throughout the entire organization. This makes it unfeasible to take the systems with potential evidence off-line, especially when considering the frequency at which network based attacks actually occur. Since syslog entries are traditionally duplicated on a central repository, the syslog facility provides a means by which the evidence can be collected without taking systems off-line, assuming of course the syslog files can be made to be legally admissible.

Computer forensics, a relatively new field of research, needs a method with appropriate authentication mechanisms in place by which syslogs can be used as relevant evidence in court. Syslogs, which have been designed more from an event logging perspective than an evidence-oriented one, are system treasure maps that chart and pinpoint attacks and attack attempts. More importantly, syslogs have primarily remained what they originally were—insufficient and cryptic [7]. Over the past few years, research on securing syslogs has yielded enhanced syslog protocols that focus primarily on tamper

prevention and detection. However, many of these protocols, though effective from a security perspective, are inadequate when forensics needs comes into play.

Over the past years, system log research has focused on securing syslogs and has advanced a great deal [8] [9] [10]. However, syslog security research cannot validate syslog entries or vouch for their authenticity with regard to the time of their creation. Therefore, what is needed is a mechanism that can validate every entity associated with a syslog entry for the log entry to be forensically viable.

The BSD syslog protocol documents one of the fundamental tenets of the syslog to be simplicity. With this as the basic focus, UDP is used [2] to transmit syslogs generated by the systems on the network to a centralized logging system. The simple format of the syslog entry and the vulnerable protocol used to transmit it makes syslogs very weak evidence; this is mainly because the integrity of syslog entries can be challenged [3].

The goal of the research presented here is to create a forensically viable syslog facility. There exists a fine difference between secure syslogs and forensically viable ones. Security research on logs has focused on securing audit logs and protecting them from intrusion and malicious manipulations. To the best of our knowledge, no research has focused extensively on making syslogs forensically viable. This essentially entails the validation of syslog entries as they are created as well as providing resistance and the detection of modifications and deletions.

2.2 Background of the Proposed Method

Every computer-based activity on a system typically leaves an electronic trace [6]. The level of understandability provided by these traces and the credibility offered by them depends on the level of security in place on the system. Electronic traces in

verifiable forms can be considered as digital evidence. In order to verify system log files, we must ensure that the log files are resistant to deletions and modifications; i.e., it may not be possible to prevent truncation of a log file, but such modifications must be detectable. Additionally, further verification must be added to the syslog protocol to validate where the syslog entries came from. Specifically, this is done using system fingerprints, user fingerprints, and application fingerprints.

In this research, we propose a new electronic trace by using a modification of the Needham Schroeder protocol [11]. The secure transmission of system fingerprints, user fingerprints, and application fingerprints is ensured by using a modification of the Needham Schroeder protocol. This protocol was developed to secure communication between two hosts by the use of session keys, random numbers, and nonces. In this method, the session keys are replaced by public keys for each system on the network. We term the public keys assigned to every authentic system K_{System}. Similar to the original protocol, these keys are generated pseudo randomly at the authentication module and are assigned to each of the systems. The weakness of the Needham Schroeder protocol lies in the use of timestamps. In the originally suggested protocol, timestamps were used explicitly. The use of timestamps explicitly enables the manipulation of messages by changing the network clock and manipulating network latency. However, this is eliminated in our proposed version due to the use of digital fingerprints, which are hashed values of various system parameters and timestamps.

2.3 Previous Syslog Research

Syslogs, developed as a UNIX protocol, are essentially a means of keeping track of events that occur and processes that run on a system. Syslogs are essential, but vastly

insufficient and cryptic. This is because syslogs were not designed from the perspective of being used as evidence or for backtracking an attack.

Waters et al. [10] present a searchable and secure audit log using asymmetric key encryption. However, this paper merely tackles the problem of storing syslogs and providing an efficient mechanism for searching through them. The paper does not have any mechanism in place to verify that the entries are generated by validated systems and have indeed been created by systems within the secure domain. An analysis after an attack would not yield sufficient evidence if this method were used. Linear hash mechanisms that detect log tampering attempts have been suggested [9]. Ayrapetov et al. [8] provide techniques that secure a syslog database using passwords. Again, this technique lacks a mechanism to control or prevent attacks that can be carried out to manipulate the syslog database. Both these papers fail to present an analysis of attacks against their proposed systems.

The approach in [12] presents the use of four entities—a generator server, a storage server, an analyzer server, and a sign server. This approach describes a secure infrastructure that signs the generated logs and stores them securely. However, this paper does not go into probing the forensic aspect of the method and how the logs secured by this technique can be proven to be generated by a valid and authenticated source.

The method presented in [13] makes use of syslogd [14] and uses the SSH package to forward logs to the server with encryption and authentication. Since this method has been mainly designed to ensure secure log transmission, validation and authentication in the event of an attack and measures to prevent the same were not explored; i.e., there is no security measure enforced that can detect log tampering. Another architecture discussed

in [15] proposes the use of IPTables to formulate rules that will limit UDP traffic to port 514, which is the port designated for syslog servers to run on. This method again does not define any kind of resilience against attacks or tampering attempts. This method at best simply defines a logging system that prevents any kind of attacks against the logging server using SSH connections and permitting communication only with certain limited IP addresses. There is no defense against modification of the log file should the system be validly or invalidly accessed.

Other papers propose logging architectures specifically from the forensic point of view for use in criminal investigations. In their research, Jiqiang et al. [16] present a schema that describes a secure logging architecture from a forensic viewpoint. It describes the entire architecture as a collection of interconnected modules, namely, host, network, receiving, classifying, and secure. However, although the aim suggests securing audit logs for use in forensic analysis, the method presented by the authors does not get into the nitty-gritty of validating log entries and the manner in which they will actually be scanned for their authenticity or tested for their genuineness. The authors' suggestion for the use of automated tools and data mining for analyzing the logs cannot really be considered as an effective scheme for verifying a syslog entry for forensic evidence due to the large number of false positives and false negatives data mining techniques are typically known to generate.

The technique Snodgrass et al. [17] present for securing audit logs incorporates a database management system to store logs, a cryptographically strong one-way hash function to secure them, and a "validator" to judge if tampering has occurred. However, this method does not focus on making logs viable forensically. Although this method

provides a means for securing audit logs, it does not provide for validating the authenticity of the source, of the transaction, or of the user that generated a particular audit log. The opportunistic hashing technique used in their research is primarily a database centric technique applicable to securing transaction records [17]. A single attack on the database storing these records will result in the loss of every audit log. Given that syslogs contain the greatest amount of data relevant to an attack, it will be a primary interest for manipulation by an attacker.

2.4 Weaknesses of the Syslog Protocol

The weakness of the syslog protocol [3] lies in the fact that it uses the user datagram protocol (UDP), a connectionless and unreliable protocol, stores system event information in plain text format, and transmits system event data across the network in plain text format. With regard to the three components of security—authenticity, confidentiality, and integrity—syslogs can be manipulated by a malicious insider or an outside attacker by exploiting these inherent weaknesses. Thus, all three components expected of security can be violated.

2.4.1 Compromising the Authenticity of Syslogs

Syslogs typically contain an immense amount of information about network and system activities. The immense size of syslog files, which is a valuable repository of evidential information, becomes a vulnerability. Syslog entries are stored independent of each other; i.e., there is no systematic chaining of log entries in a syslog file. Log entries are, in fact, independent of every other item in a particular log file.

Additionally, there is no relationship between the system and the facility that generates a syslog entry. Further, no authentication mechanism exists by which syslog

entries can be validated and be claimed to have originated from one system and one facility. By exploiting this, an attacker can send random and spurious entries to the syslog file by spoofing source addresses, which could be either completely incorrect or spoofed from a legal and authentic system. Tools such as netcat, crypt-cat, etc., can be used to carry out this spoofing. In the event that several attackers carry out such planned flooding in parallel, the impact would be sufficient to cause a denial of service attack against the syslog server. Since syslogs seldom have a dedicated server, this kind of attack will also bring down other applications that reside on the same server.

2.4.2 Compromising the Confidentiality of Syslogs

Syslogs are a lucrative source of evidence of electronic activities that happen in an organization. In spite of this, as originally developed, syslog entries are still transmitted in plain text and are even stored centrally in an unencrypted form. With the ease at which open-source network tools are available off the Internet, a readily available tool such as tcpdump can be used to sniff syslog entries being transmitted to a central logging repository. By sniffing and analyzing these entries, an attacker, aside from attacking the systems themselves, can determine precisely how to inject messages into the syslog file.

2.4.3 Compromising the Integrity of Syslogs

Syslogs that are stored on a central logging repository are accessible to only the root user or the system administrator. An attack on the central repository and the procurement of root access enables access to all the system logs. The logs are then open to one or more of the following attacks—multiple entry deletion, malicious modification, abrupt truncation, or complete deletion. Furthermore, UDP traffic can be sniffed, replayed, and manipulated, thereby making syslog entries highly questionable. Attackers will often

delete entries related to their activity to avoid detection. Simultaneously, this prevents the syslogs from being effective forensic tools for legal admissibility.

2.5 Forensic Requirements

Dixon [18] identified the primary characteristics that computer forensic evidence entails. These include:

- Preservation
- Identification
- Extraction
- Documentation
- Data Interpretation
- Confidentiality
- Integrity
- Availability

These are discussed in detail below as well as how our proposed techniques more fully fulfill the requirement. The goal of our research is to integrate more of these requirements than has traditionally been done. For instance, while secure log files add a level of identification, they are greatly lacking in terms of evidence identification, preservation, and extraction. Our proposed technique attempts to fulfill these first three requirements for forensic viability while maintaining the confidentiality and integrity provided by recent research in secure log files.

Bishop [19] specifies that any secure system needs to safeguard the following three components: confidentiality, integrity, and availability. A compromise of any of these

components will render the system as insecure. A syslog is secure if it maintains its confidentiality, integrity, and availability. In terms of forensics, data will generally not be forensically viable if the system collecting and storing the data is not secure as a starting point.

2.5.1 Preservation

In our technique, the use of user fingerprints, application fingerprints, and system fingerprints validates all the entities involved in the generation of a single syslog message. The authentication traces stored on each system provides sufficient information to backtrack an event that might have occurred.

2.5.2 Identification

Authentication traces can be exemplified as local, simpler, copies of syslog files with the difference that they contain fingerprints and timestamps. When an incident occurs and syslogs have to be analyzed, the local copies of the authentication traces can serve as additional evidence to back up the facts presented by the central system log entries.

2.5.3 Extraction

The authentication traces that belong to a particular system must be stored in an encrypted format. Thus, only system administrators would have the privilege to decrypt and read the locally stored authentication traces. This greatly limits attacks, as individuals will not know what they are attempting to attack. For instance, attempting to inject events is difficult without knowing the contents of the files. Similarly, access must be limited to read-only to limit the potential for modification. In general, system log files should only be appended to in order to limit their susceptibility to attack.

2.5.4 Documentation

The chain of custody after the acquisition of digital evidence ought to be documented correctly.

2.5.5 Interpretation of the Data

Computer evidence is typically not in a human-understandable form. In order to elicit appropriate responses from the jury, when digital evidence is very technical, an expert witness is required to interpret these results in a court of law. The authentication traces in our proposed method can be used as evidence to reinforce the prosecuted claims. Computer forensics is a two-stage process that typically comprises:

The method presented in this paper tackles the first stage of this process.

2.5.6 Confidentiality

Confidentiality refers to the concealing of a resource or a system from entities that "do not have the need to know" [19: 4]. The read access right for syslog files essentially belongs to the system administrator who has root privileges. The administrator should not be granted privileges to "write" to the syslog, regardless of whether it is stored locally or centrally, because this would defeat the very concept of a syslog being a log of events as they happen. This can be enforced through:

- Well-defined access control rights for system users
- Password files encrypted and not stored locally
- Encrypting syslogs
- Remote logging
- Modifying the location of the logging host in the syslog.conf file

2.5.7 Integrity

With respect to a secure syslog, integrity refers to the trustworthiness of the data it contains. It also refers to the integrity of the entities that generate the log entry, the integrity of the medium that transmitted the entry, and the integrity of the system that actually stores the data. The information presented by the syslog files should be accurate and should be trustworthy enough to be used as evidence by a forensic expert or an administrator. A syslog file that contains spoofed or tampered entries is not forensically viable. Integrity mechanisms fall into two categories: prevention and detection.

Prevention mechanisms seek to maintain the integrity of the data by jamming any unauthorized attempts to access data and modify it in unauthorized ways. A more challenging task would be to prevent an authorized user from modifying the data in unauthorized ways. Strict authentication mechanisms on the host and the server can help enforce this kind of integrity check. More importantly, if remote logging is indeed being used, ports on the logging server should be filtered appropriately.

Detection mechanisms on the other hand do not try in any way to prevent intrusions into the system or in any way to safeguard the integrity of information stored on it. Instead, detection mechanisms simply identify and log all accesses. Particular attention is paid to identifying who made specific access, when the accesses occurred, and what was done during the access.

Most contemporary systems today incorporate characteristics from both prevention and detection. Essentially, the system uses access authentication to limit access but goes under the assumption that no prevention technique is 100% accurate and thus also records all accesses as per strict detection mechanisms.

2.5.8 Availability

Availability refers to the availability of syslog data when needed. If they have adequate information on log transmission, attackers can launch a denial of service attack with the goal of preventing the central repository from receiving event entries.

2.6 Syslog Variants

In our goal to develop new techniques for creating forensically viable syslog facilities, we examined existing capabilities to identify what existing work we could draw from and build upon, rather than doing the entire research from scratch. The two existing systems dealing with secure syslog facilities that offer the greatest capability on which to build are syslog-sign and syslog-auth. Other variants such as syslog_reliable [20] and syslog_ng [21] do not provide any form of forensic credibility.

While many of the below mentioned capabilities provide improved validation and authentication from a security perspective, these improvements are insufficient for forensic validity, i.e., for legal admissibility. These existing capabilities are not sound enough to prove beyond a reasonable doubt that an attack occurred and the characteristics of that attack. Extending current capabilities to this level is the goal of our research.

2.6.1 Syslog-Sign

This protocol defines three types of messages: normal messages, signature blocks, and certificate blocks. It typically transmits to the central repository a signature block after a certain number of syslog message packets have been transmitted [22].

The limitations of this system include the ability for an attacker to flood the syslog server with plausible-looking messages, signature blocks, and certificate blocks [23]. Since the number of messages after which a signature block is generated is fixed, a wily

attacker can eliminate the very presence of these signature blocks. This protocol warrants the online construction of hash tables, which increases overhead costs.

2.6.2 Syslog-Auth

This version of syslog uses a shared-key principle. It works on the basis that the syslog packets are encrypted at every hop using the keys of the previous sender and the current recipient. The auth block comprises several blocks. Each syslog packet is parsed from the beginning to the end of a block. This protocol is more suited for an online analysis and, hence, is better than Syslog-sign [23].

The limitation of syslog-auth is a result of the fact that the key management is a challenge since every device and relay has its own key. Further, the routing of messages through different relays further complicates key management. Since an attacker knows that the auth block is appended to a syslog message, the attacker can rip off the block entirely, thereby destroying the authentication mechanism that the syslog-auth actually provides. Finally, this protocol does not provide for origin authentication or validation.

CHAPTER 3

THE PROPOSED METHOD

3.1 Overview of the Proposed Method

Figure 3.1 shows an overview of the system proposed in this paper. Currently, in order for syslogs to be worthy of being considered as evidence in forensics, what is needed is an authentication mechanism that reinforces and authenticates what the system log file presents. The entities involved are the user, the application, the system, the client syslog daemon, the authentication module, and the syslog server. The client syslog daemon and the syslog server are not shown explicitly in this overview diagram.

In our proposed protocol, there are two servers, an authentication server and a logging server. The authentication server records every authentication that occurs and maintains their timestamps. Since this server needs to act as a form of backup in the event that

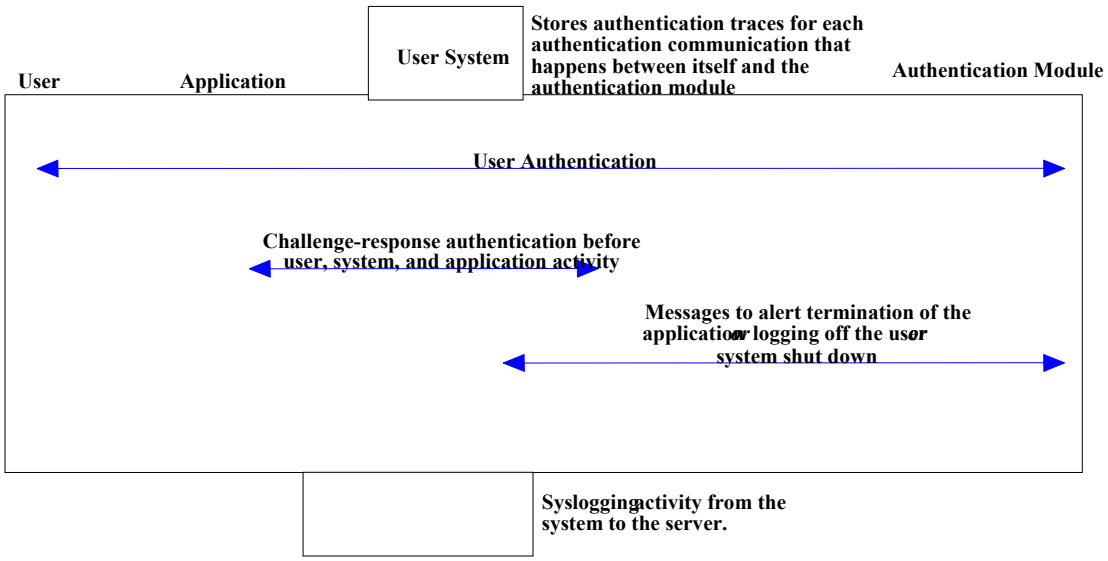

Figure 3-1. View 1 of overview of the proposed method.

system logs on the logging server are tampered with or additional evidence is needed to verify a claim, it will have a minimum number of processes running, limited accessibility, and constrained resource availability. Further, this server can decipher the entries in the individual prints and verify the authenticity of a fingerprint. The logging server stores actual log entries and is the main storage system for these log entries.

In addition to the background processes of syslog generation and authentication trace generation, which are umbrella processes that exist throughout a session, the proposed approach comprises three main active steps.

User authentication: This is based on desired login authentication procedures and is geared toward ensuring that only authorized users access the system. The user is authenticated by the server.

Challenge response before the user, system, and application become active: This step encapsulates and comprises the generation of user fingerprints, application fingerprints, and system fingerprints. Furthermore, in order to cement and secure the transmission of these fingerprints and the authentication traces, which are generated by individual systems, several challenge response steps have been incorporated.

Messages log the termination of the application, logging off of the user, and the shutting off the system: This is an authentication mechanism primarily focused on ensuring that the same entity that has been granted login privileges has logged in and is the entity sending event messages. However, with regard to computer forensics, a mechanism to verify the termination of an authorized entity is also needed. This step details a secure and logged termination of the entities involved in the generation of a syslog entry.

3.2 The Proposed Method

Our proposed protocol, exemplified in Figure 3.2, proceeds through the following six phases:

1. User authentication
2. System connection establishment
3. System connection establishment response
4. Application event entry generation
5. Applications termination
6. System connection termination

3.2.1 Phase 1: User Authentication

This step uses the basic credentials that a user needs to log onto the system—their user name and password. The authentication module verifies the authentication pair and sends back an acknowledgment.

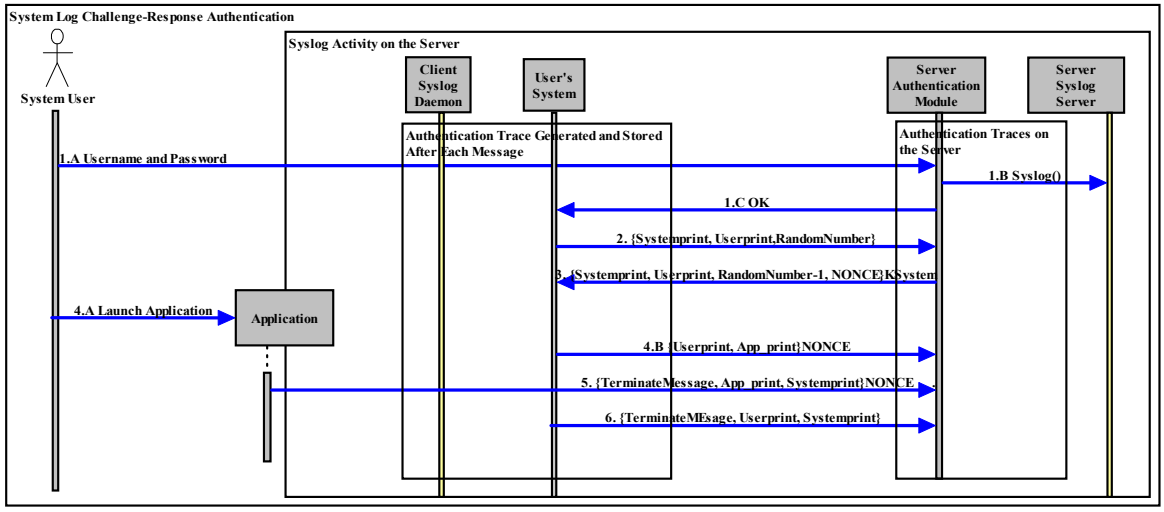

Figure 3-2. View two of overview of the proposed method.

3.2.2 Phase 2: System Connection Establishment

$\{\text{systemprint, userprint, randomNumber}\}_{\text{KSystem}}$

The systemprints and userprints are used to establish the fact that a particular system has logged onto the network and is being used by a specified user. The randomNumber is used to emphasize the one-time nature of the communication. A validation mechanism is in place on the server to verify randomNumbers and catch suspicious duplications of the random numbers, if any, i.e., the random numbers should not be reused. The authentication server would notice that that the particular random number has been used already and more importantly, it has replied to the message.

Since, by definition, we cannot guarantee the uniqueness of random numbers, they are not used in isolation. Even if the random numbers are repeated, the authentication streams (here, digital fingerprints) that they are used to create will still be unique. This is because the fingerprints, as previously stated, are a function of several parameters and a random number is just one of them. More importantly, even if random numbers happen to be repeated, the streams that they are a part of, namely, the fingerprints and the challenge response, will be unique.

3.2.3 Phase 3: System Connection Establishment Response

$\{\text{systemprint, userprint, randomNumber-1, NONCE}\}_{\text{KSystem}}$

This message is sent in reply to the connection establishment message sent by the client. The use of the nonce here signifies the one-time nature of the communication. If an intruder sniffed this message and tried to replay it, the replayed message would have no consequence on the network, and would in fact identify the presence of the intruder.

The randomNumber is the same as the one sent initially by the client. The nonce functions as a kind of a one-time key to be used by the users.

3.2.4 Phase 4: Application Event Entry Generation

$\{\text{userprint, app_print}\}_{\text{NONCE}}$

The nonce is used to prevent any form of man-in-the-middle attack. The key used is the nonce transmitted by the server in the previous communication. The one time nature of the nonce prevents an attacker from launching a man-in-the-middle attack since the key is generated for each system uniquely and is meant to be of a one-time nature.

3.2.5 Phase 5: Applications Termination

$\{\text{terminatemessage, app_print, system_print}\}_{\text{NONCE}}$

This message logs the actual termination of an application. In order to be able to validate information forensically that can be used as evidence, it is necessary to be able to validate the time at which an application has been terminated. Events received after application termination would be indicative of an intruder or compromise.

3.2.6 Phase 6: System Connection Termination

$\{\text{terminatemessage, user_print, system_print}\}_{\text{NONCE}}$

This message is sent by the client system when either the system shuts down or the user logs off. This again aids in validating event entries and limits the ability of an intruder to compromise the log reporting facility.

CHAPTER 4

FIINGERPRINTS AND AUTHENTICATION TRACES

In physical forensics, fingerprints are one of the key factors that reveal evidence about the perpetrator or identify key entities (people or objects) involved in a crime. Creating digital versions of fingerprints of every entity involved in the generation of a syslog entry promotes and emphasizes the need to make every entity responsible for ensuring its forensic viability.

4.1 User Fingerprints

User fingerprints tightly bind the user and the system used. The user print can be considered as simulating a real life fingerprint. When a fingerprint is considered in the real world, factors such as location and time are also taken into account before arriving at conclusions. Thus, for the cyber version of user fingerprints, similar types of information must be included, i.e., user identifying characteristics, time, and system identifying characteristics. This ties a specific user to a particular system at a specific time. More specifically, we propose using the following to create a user fingerprint:

- Username and password
- System MAC address
- Login time

Clearly, much of this information could be individually compromised or improved upon. However, the compromise of these individual components should be identifiable. Additionally, should the resources be available, more secure paradigms can be used. For

instance, the Air Force requires use of physical access cards to log into any computer system that would provide greater integrity than usernames and passwords alone.

Attempts to compromise the individual components would fall back onto typical computer security paradigms. For instance, the server should identify the fact that the time used by the client system is unacceptably out of scope with the server's time. A compromise of the system MAC address would be identifiable through duplicate MAC addresses, the change in router paths to the MAC address, or detection of an invalid mac address.

4.2 Application Fingerprints

Application fingerprints are similar to user fingerprints. The application fingerprint will be generated for every application that is launched on a system. Their primary role is to identify and distinguish between legal applications and illegal ones launched by specific users on a system. As with user fingerprints, the goal is to provide as much identifying information as possible. In this case, we are attempting to validate what application is being run, by whom, when, and from where. Thus, application fingerprints would use the following pieces of information:

- Launch time
- Username
- System mac address
- Application identifier

In a large system, every application on the system would have a different application identifier. In our current view of the model, application identifiers are generated on the fly, and the identifiers that are generated for each application are documented. As with

system connection establishment, the randomly generated application identifiers are not used in isolation due to the lack of guaranteed uniqueness of the identifiers. Further, when application IDs are logged in authentication traces, they have the application name logged with them as well to aid in differentiation.

4.3 System Fingerprints

System fingerprints are often used by operating systems manufacturers to register the system on which the operating system was installed and ensure it is not transferred to a new system in violation of the operating system license. The concept of system fingerprints essentially relies on the fact that once deployed most systems rarely have their configuration change, especially in business environments. For home users, while some sophisticated users upgrade individual components of their system, the majority of home users will not. Many different characteristics can be used to identify a system uniquely. Some possibilities include:

- The number of processors
- Disk space
- System mac address
- CPU ID
- Installed applications
- Disk drive identifier, serial number

We have actually found that each individual hard drive has a unique serial number that is installed in the hard drive bios that is generally read only and is accessible using free programs available on the net [24]. This identifier should prove to be particularly effective as a system fingerprint.

4.4 Fingerprint Generation

User fingerprints, application fingerprints, and system fingerprints are generated using the RS hashing algorithm, which is known to have low collision rate. The RS algorithm, which is a general-purpose hashing algorithm developed by Robert Sedgwick [25] is used to generate hashes, i.e., fingerprints.

Sedgwick's hashing algorithm is a rotative hashing algorithm that uses rotative hashing. In rotative hash functions, unlike its counterpart, the values are bit-shifted. Sometimes combinations of both right and left bit shifts are used. For increased security, bit shifts are sometimes prime numbers. The intermediate value that is yielded at each step is added to an aggregative value. The result that is yielded is the value of the final aggregation. An example:

$$hash = hash^{-1} \oplus (t << p) \otimes (t >> q)$$

4.4.1 User Fingerprint Generation

For the user fingerprint, the key is a concatenation of the username, the time of user log in, and the user ID that was generated when s/he logged in. Keys in a hash function are required to be unique so as to avoid collisions and enable faster look up. The keys here are concatenation of three parameters that will most certainly be unique across logins in an organization.

The algorithm is coded as shown in Figure 4-1. However, different keys are used for the user, application, and the system fingerprints.

4.4.2 System Fingerprint

The system fingerprint is generated in the same way. We have found that the hard disk serial ID that is hardcoded by a manufacturer is the only unique parameter than can actually distinguish one system from another. The hard drive serial IDs, which are assigned to every partition on the hard drive, were another parameter that was considered. However, these IDs can be changed when the disk is reformatted. Another parameter that was considered was the CPU ID. A run of an application on laboratory systems revealed that all CPU IDs that belong to computers ordered in bulk are the same. The MAC address was not considered as a potential parameter due to the ease by which a person with reasonable computer knowledge can change and even spoof a MAC address. The key used in this case is the hard disk serial ID. This was identified and verified to be

```
for(int keyLength=0;keyLength<fingerPrintKey.length();keyLength++){
   long intermediateUserChar = (long) fingerPrintKey.charAt(keyLength);
        fingerPrintH = (fingerPrintH << 4) + intermediateUserChar;
        fingerPrintG = fingerPrintH & 0xF0000000L;
        if (fingerPrintG != 0)
              fingerPrintH ^= fingerPrintG >>> 24;
        fingerPrintH &= ~fingerPrintG;
}
return (long)(fingerPrintH);
```

This user print yields→ 155990563

Figure 4-1. User fingerprint generation.

unique. Therefore, the hard disk serial ID and the system bootup time are together used as a key for generating the system fingerprint. The system print is generated as shown in Figure 4-2.

4.4.3 Application Fingerprint

The application fingerprint is necessary in order to validate applications. Here, a concatenation of the username, application ID, and the applicationTimeStamp is used as the key in the fingerprint generation (see Figure 4-3).

The extent of this implementation is the generation of the authentication traces, digital fingerprints, and the simulated syslog entries. The implementation was carried out with the aim of deriving a prototype of the proposed method. The challenge response mechanism was incorporated as part of the remaining implementation.

```
systemFingerPrint(){
   String HDDSerialNumber= "97LET9BET";
   String systemFingerPrintKey=
HDDSerialNumber.concat(systemBootupTime);

   for(int keyLength=0;
keyLength<systemFingerPrintKey.length();keyLength++){
     long systemIntermediateChar=(long)
systemFingerPrintKey.charAt(keyLength);
          systemPrintH = (systemPrintH << 4) + systemIntermediateChar;

          systemPrintG = systemPrintH & 0xF0000000L;
          if (systemPrintG != 0)
               systemPrintH ^= systemPrintG >>> 24;
          systemPrintH &= ~systemPrintG;
 }
return (long)(systemPrintH);
}
```

An example of a system fingerprint yielded by this method → 161044579

Figure 4-2. System fingerprint generation.

```
applicationFingerPrint(String applicationName, long applicationID, String appLaunchTimestamp) {

  for(int keyLength=0;keyLength<applicationFingerPrintKey.length();keyLength++){
    long appIntermediateChar = (long) applicationFingerPrintKey.charAt(keyLength);
            appPrintH = (appPrintH << 4) + appIntermediateChar;
            appPrintG = appPrintH & 0xF0000000L;
            if (appPrintG != 0)
                    appPrintH ^= appPrintG >>> 24;
            appPrintH &= ~appPrintG;
   }
  return (long)(appPrintH);}
```

An example of an application print yielded by this method→76274804

Figure 4-3. Application fingerprint generation.

4.5 Authentication Traces

An authentication trace is an entry that is generated on every system on the network and records the generation of system, user, and application fingerprints along with the associated timestamps. Authentication traces on each system can be viewed only by administrators. The traces are typically a message along with the prints and the timestamp of the event.

The RS algorithm, which is a general-purpose hashing algorithm developed by Robert Sedgwick [25] is used to generate hashes, i.e., fingerprints. A test carried out [26] shows that the RS algorithm had very few collisions when tested on a huge string data set. Examples of valid and invalid authentication traces are shown in Figure 4.4.

4.6 NONCES

A nonce is a number that is used only once. It is typically used in protocols that aim to ensure secure communication and prevent any form of man-in-the middle attacks. The transmission of the userprint and the app_print needs to be secure. For this reason, the server generates a nonce that is meant to be used only for *one* transmission of the systemprint and userprint. This prevents a replay attack. Moreover, even if an intruder sniffed it, it would be of no consequence to the network.

For every event that occurs on the system on the network, the syslog daemon creates syslog entries. The result acquisition in this research aims to be able to map back to the true user, system, and application that caused the corresponding syslog entry by using

```
steena logged in at  2007-12-30 06:46:14 with user ID 3488469706175790508     with
user finger print   88726020 The system print is 94173252

C:\Program Files\Internet Explorer\IEXPLORE.exe      launched  at  2007-12-30
```

the associated authentication traces and fingerprints.

```
06:47:26     with ID      4384844220178160764 with fingerprint    223266966


C:\Program Files\Internet Explorer\IEXPLORE.exe      terminated      at2007-12-30
06:51:13     with ID      4384844220178160764 with fingerprint 223266966
```

(a)

```
Incorrect login with username:   ghost  occurred at2007-12-28 23:01:16   with
userID 2221344687639655740with userprint 238806054
```

(b)

Figure 4-4. Example authentication traces. (a) A valid authentication trace. More specifically, the authentication trace of a user named “steena” logging in and launching Internet Explorer. (b) The authentication trace of an invalid login.

CHAPTER 5

ATTACK BACKTRACKING

5.1 Backtracking to an Attack

Syslog entries typically comprise the following parameters—hostname, facility, priority, message, and timestamp. This implementation simulated a syslog logging facility. The purpose of this was to compare an authentication trace and be able to get to the fingerprint from the syslog. After an attack occurs, parameters from the syslog can be used to obtain the corresponding entry contained in the authentication trace. An important point to be noted is that time is a crucial factor in the generation of an authentication trace and the corresponding syslog entry. The user, application, and system are the facilities considered in this implementation of syslogs. Their priorities are hardcoded here since this implementation mainly serves as an example and validation of how authentication traces, and syslog entries can be used in tandem to trace back and form evidence. The research in [27] suggests that authentication traces can be used to backtrack to an attack. The authors of [28] show this can be actually carried out. This is because every parameter that is considered in the generation of a fingerprint can be essentially obtained from the corresponding syslog entry in the log file. Therefore, the authors demonstrate the way in which attacks detected in the syslog entry can be backtracked using a combination of the authentication traces, the syslog file, and the hash function (here, the RS algorithm).

The following authentication trace shows a login by user “steena” and the corresponding syslog entry.

```
\
steena    logged in at   2008-02-06 12:49:33 with user
ID   7524389880967786033   with   user   finger   print
     155990563 the system print is161044579

C:\Program     Files\Internet     Explorer\IEXPLORE.exe
launched at    2008-02-06 12:49:41 with              ID
     1524843500148472672 with fingerprint    88504721
```

The corresponding syslog entries with format host name, facility, priority, message, and timestamp.

```
localhost 4    10   steena has logged in at   2008-02-
06 12:49:33

localhost 6    12   C:\Program           Files\Internet
Explorer\IEXPLORE.exe launched at       2008-02-06
12:49:41
```

Repeated bad logins at a particular system will yield corresponding authentication traces and syslog entries. However, the occurrence of repeated bad logins will be logged by the authentication traces and not by the system logs, unless they are configured to do so.

```
  Incorrect login with username:  steena    occurred
at2008-02-07 04:50:02   with                        userID
    56032638045929763with userprint    188996098


   Incorrect login with username: steena    occurred
at2008-02-07 04:50:28   with                        userID
    89362438861078928 18with userprint  188996200


   Incorrect login with username: steena    occurred
at2008-02-07 04:50:43   with userID
2404564924573438423 with userprint     188996163
```

An attack by a malicious insider causes the username, which is already known, to be exploited. In this simple emulation of system logs, we have explicitly logged a bad login instead of a series of repeated logins by a valid user.

5.2 Reconstructing Fingerprints

The user fingerprint comprises the username, the user ID, and the time of login. These values can be obtained from the syslog entry. A hash of these parameters using the RS function yields the corresponding fingerprint. The absence of authentication traces only reveals the persistent login by user “steena.” A closer examination of the system logs and its corresponding authentication trace can even possibly reveal the identity of the person behind the attack. A small script to check and match users who have already

logged in and their log in times can possibly reveal this. A more complex implementation aims to assign appropriate priorities and facility numbers to every entity involved in the system.

An important point to be noted while logging events to a central repository is that the local system time for each individual system should be used instead of the server time. This is because authentication traces are generated and are representative of activity by entities on those individual systems. The use of server time would lead to misinterpretation of events on those systems. This was noted during the current implementation when entries were being logged successfully but had a clear disparity with regard to timestamps in their corresponding authentication trace entries.

CHAPTER 6

FORENSIC VIABILITY

6.1 Requirements of Forensic Evidence

Previous research has dealt with using digital evidence in a court of law as documented in [29]. Forensically viable log files as defined in [30] requires that log files be created and stored by keeping legal investigation procedures in mind. The three factors to be considered when dealing with log files as evidence as suggested in [30] are:

1. *Logs must be protected against losses*. In the proposed method, the use of fingerprints as well as the generation and secure storage of syslogs ensures the integrity of the syslogs. This is done through a second source of evidence—the authentication traces.
2. *Evidence must be found within log files*. The authentication traces document the authenticity and validity of every entity and activity involved in the generation of a syslog entry through explicit messages and fingerprints.
3. *Log file information should be documented for additional judicial scrutiny* [30]. The explicit authentication traces serve as backup/reinforcing evidence for syslog entries. These traces contain copious amounts of validating and authenticating information in a succinct form.

6.2 Evidence Certainty Levels

The research in [29] assigns predefined levels of certainty to digital evidence collected from affected systems with C0 having the least certainty and C6 the highest certainty.

Digital evidence needs to have a degree of certainty attached to it in order to make it credible, and thus for it to be legally admissible or accepted by a jury. A mapping of these levels of certainty to syslog files is presented in Table 6-1.

Table 6-1. Mapping the Certainty Levels Defined in [2] to Syslog Files.

Level	**Level Confidence**	**Relationship to Syslogs**
C0	Erroneous/ Incorrect	Programmatic errors while coding the syslog/sylslogd protocol [3]. An attack occurs by exploiting this vulnerability.
C1	Highly Uncertain	A patchy syslog file with manipulated entries.
C2	Somewhat Uncertain	In the event of an attack, the only evidence that is available is the organizational syslog file. Distributed evidence preservation—proposed in this paper—has not been attempted.
C3	Possible	Syslog variants, namely, Syslog-sign and Syslog-Auth, have this level of certainty.
C4	Probable	Syslogs and authentication traces that are stored and transmitted in plain text can be classified to have this level of certainty.
C5	Almost Certain	This level of certainty specifies evidence to be tamperproof and asserts a match between independent sources of evidence, which in this case are the authentication traces and syslogs. The evidence at this level, however, can be erroneous due to temporal loss or data loss. The currently proposed method belongs to this level of certainty.
C6	Certain	If authentication traces were validated at every system that they were generated on and more importantly, at intermediate stages in the routing to the syslog server, syslog evidence would then have this level of certainty.

According to Table 6-1, the evidence presented by the syslogs, which was collected and generated by our approach, falls into the C5 level, given the authentication and validation capabilities integrated into the model. On the other hand, typical syslog capabilities and even secure syslog facilities achieve a much lower ranking. The log files generated by this method can be termed as forensically viable as defined by research in [30].

6.3 Use of Authentication Traces and Syslogs Under Certain Scenarios

Authentication traces and syslogs can be used in other circumstances other than backtracking an attack, which of course, is its primary aim. The three scenarios below exemplify some of these characteristics. For these scenarios, consider the fictitious entity SecurityVille. SecurityVille is an organization in which every user has a dedicated system and a login username and password. Andy is the administrator; Fred is the forensic analyst; Steve is the malicious insider, who is also an employee; William is a wily external attacker; and Arby is another employee. Authentication traces are maintained on every system and on the server. Syslogs are maintained only on the server.

6.3.1 Scenario One: Syslog File Deletion

The SecurityVille network has been taken offline due to an attack by William. Knowing the immense repository of information that syslogs contain, Fred begins searching for the syslog file on the server. However, William, knowing this too, has deleted it.

The authentication traces serve as complimentary evidence. Although the fingerprints are indecipherable at a glance, a further inspection of the authentication

traces can yield an almost complete reconstruction of the syslog file, thereby showing the origin of the attack, its modus operandi, and to a limited extent the severity of the attack.

6.3.2 Scenario Two: Spurious Entry Injection into the Syslog File

During a fortnightly inspection of the syslog file, Andy notices that certain entries appear to be invalid, i.e., not matching the authentication traces. Clearly, someone has managed to alter the syslog file on the server. The corporate network logs, router logs, and switch logs do not reveal any suspicious activity. As it happens, the attack originated from an internal source: Steve has managed to gain access to the server and injected spurious entries into the syslog.

An inspection of his authentication trace reveals that he has managed to install a rogue application on his system. His traces reveal the name of an unknown application.

6.3.3 Scenario Three: Application Updates

FortyTwo, which is an accounting software used by the employees, is scheduled to undergo updates every two weeks.

In the method proposed here, before an application launches, it needs to go through the challenge response mechanism. The application fingerprint is then calculated on the fly. When the application has been updated and has to restart, its print is recalculated and the restart is treated as the launch of a new application. Since application IDs are assigned on the fly and are documented, the automatic updates would not affect the generation of the application prints and their transmission. Currently, the authentication trace generation has no mechanism to determine if an update has occurred or if the user has merely chosen to close and launch the application again. However, a close examination of the traces across systems and the system logs would reveal this update if a pattern of

restarting an application is seen across multiple systems. Further, since the application name is listed in the authentication trace, this pattern will be readily found. An application update occurring while the application is not running would not lead to any suspicious traces, the desired result.

CHAPTER 7

PROTOCOL RESILIENCE

7.1 Attacks Against the Challenge Response System

We have previously discussed some of the weaknesses of the syslog protocol and the ineffectiveness of secure syslog facilities for use when forensic viability is required. Here, we discuss the specific capabilities of our proposed model and the resilience these capabilities provide against typical attacks that are not handled by typical syslog facilities.

7.1.1. Phase 2: System Connection Establishment

An intruder can easily sniff this phase's message. However, the intruder cannot replay it because the intruder would need to authenticate to the particular system from which the intruder wishes to launch the attack. It is only after a user is authenticated and a user fingerprint generated that this communication can be initiated. The user fingerprint contains parameters known only to that user which limits the potential for compromise.

7.1.2 Phase 3: System Connection Establishment Response

Even if the message associated with this phase is sniffed, it is unreadable since it is encrypted and can only be decrypted by obtaining the private key from the system. Moreover, this is a response message, replaying it would not cause a successful attack since it would require the previous authentication and connection establishment steps to be completed successfully.

7.1.3 Phase 4: Application Event Entry Generation

The nonce used to encrypt this message is generated by the server and is sent to the system via an encrypted transmission. Even if the intruder replays this message, it will be detected as a replay attempt due to the presence of the already-used nonce.

7.1.4 Phase 5: Applications Termination

At this stage, a sniffing attack will fail because of the encryption using the K_{System}. More importantly, Nonce2 is generated only when the previous challenge communication is met.

7.1.5 Phase 6: System Connection Termination

Man-in-the-middle attacks fail since all the entries will be encrypted using Nonce2. If an intruder needs to determine the entries and the prints, the intruder will need to sniff out Nonce2, which is sent via an encrypted communication.

7.2 Attacks Against the Syslog File

The truncation of syslogs is currently a major issue. However, here the authentication server maintains logs of every authentication and challenge response that has occurred. Truncation in this case succeeds only in deleting the explicit entries generated by the systems on the network. The attacker would not be able to delete the trace in the form of authentication server logs that point to the entities and the authentication mechanisms involved in the generation of those entries. For example, deleting a chunk of successive entries from the logging server does not eliminate the fact that a certain event had occurred since the logs on the authentication server still have evidence of every communication that has occurred.

7.3 Denial of Service

This type of attack floods the server and consumes available resources in an attempt to disrupt logging activity. More importantly, if authentication and logging are not separated, this attack has a better chance of being successful. In our proposed model, proper syslog entries (those that report actual events on a system) are not generated until a proper authentication is accomplished. Therefore, neither the syslog server nor the authentication server allocate resources or even log any entries before a proper challenge-response authentication can succeed. This protocol is therefore resilient against the denial of service and flooding attacks—two very frequent attacks.

7.4 Abusing Privileges

A trusted user can abuse existing privileges and bypass protection mechanisms to gain unauthorized access to the logging server and to the log entries themselves. In our approach, every user on the network is required to authenticate to the server using an authentication mechanism followed by a series of challenge response authentication. This authentication mechanism does not permit any kind of unauthorized write attacks. The write attack is eliminated on account of the user prints, application prints, and system prints associated with every entry. Additionally, the server is designed to be appended to only; any other modifications to the log file, insertions, deletions, etc. are considered attacks.

7.5 Application Updates

In the method proposed here, before the application launches, it needs to go through the challenge response mechanism. The application fingerprint is then calculated on the fly. When the application has been updated and has to restart, its print is recalculated, and

the restart is treated as the launch of a new application. Since application IDs are assigned on the fly and are documented, the automatic updates do not affect the generation of the application prints and their transmission. Currently, the authentication trace generation has no mechanism to determine if an update has occurred or if the user has merely chosen to close and launch the application again. However, a close examination of the traces across systems and the system logs would reveal this update if a pattern of restarting an application is seen across multiple systems. Further, since the application name is listed in the authentication trace, this pattern is readily found. An application update occurring while the application is not running would not lead to any suspicious traces, the desired result.

CHAPTER 8

CONCLUSION

8.1 Current Scenario

The proposed method provides a mechanism to authenticate and validate syslogs. Although syslogs have been researched extensively from the security perspective, they have not received sufficient attention from the forensics point of view particularly for ensuring legal admissibility. The fingerprints assigned to every entity involved in system log generation enables the validation of these entities. More importantly, since digital evidence is treated in the same way as documentary evidence [31], a means to authenticate and verify its authenticity is needed. The proposed method provides resilience against common attacks launched against syslogs—system log truncation and man-in-the-middle attacks, which are currently of the most significant problems, associated with using system logs as evidence in court. For instance, the credibility of system log files as evidence can easily be attacked in court and invalidated.

With the proposed method, suspicious activity by a malicious insider can be traced back to him/her. His/her system identity can then be forensically verified by hashing the values available in the syslog file and the authentication traces, using the RS algorithm, and matching them with the prints in the authentication traces. This mechanism is limited to tracing back to insiders only. The ability to trace back to an outside attacker is beyond the scope of our proposed method, though the internal compromised identity would be identified.

An important aspect to be noted is that the proposed method clearly satisfies the following authentication requirements of electronic evidence documented by the

Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice [5]:

1. Verification of the authenticity and prevention of the alteration of computer records

   This is achieved through the use of the authentication traces and the challenge response mechanism.

2. Establishment of the reliability of computer programs.

   This is achieved through the use of application fingerprints.

3. Identification of the author of computer-stored records

   This is achieved through the use of user fingerprints.

8.2 Future Work

An extended implementation of this method should enlarge the prototype developed in this research to for a wider implementation that uses the actual syslog daemon and a central logging repository. In addition, the development of a sophisticated authentication module would help realize several security features and satisfy several requirements, which has been proposed in the above sections.

With forensic trace back as the prime focus of this method, the important feature that should be focused on is that of successful attack backtracks. The accuracy of each attack backtrack needs to be measured and tracked. This metric is of paramount importance due to the sensitive nature of forensic evidence and the fact that this evidence can be used as crucial evidence to incriminate the culprit.

In addition, the resilience of this method against the various attacks listed in Chapter 7 should be tested. The strength of the communication between the individual systems on

the network and the servers lies in the security of the challenge response mechanism proposed. The list of attacks mentioned in Chapter 7 is not exhaustive and should be built upon as work on this method progresses.

REFERENCES

[1] Forensics International. Computer evidence processing good documentation is essential. 2008. http://www.forensics-intl.com/art10.html April 2008.

[2] Lonvick, C. RFC 3164-The BSD syslog protocol. Cisco Systems, August 2001.

[3] Nawyn, K. E. A security analysis of system event logging with syslog. SANS Institute, 2003.

[4] LogAnalysis.org. 2006. http://www.loganalysis.org/sections/parsing/generic-log-parsers/. April 2008.

[5] Cybercrime.gov. Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations. 2002.

[6] http://www.cybercrime.gov/s&smanual2002.htm April 2008.

[7] Volovino, L. Electronic evidence and computer forensics. *Commun. Assoc. for Information Systems 12* (2003), 457-468.

[8] Peisert, S. Forensics for network administrators. *USENIX* (2005), 34–42.

[9] Aryapetov, D., Ganapathi, A., and Leung, L. Improving the protection of logging systems. Technical Report, UC Berkeley, 2002.

[10] Schneier, B. and Kelsey, J. Secure audit logs to support computer forensics. *ACM Trans. on Information and System Security 2*, 2 (1999), 159 - 176.

[11] B. Waters, Balfanz, D., Durfee, G., and Smetters, D.K. Building an encrypted and searchable audit log. In *11th Annual Symposium on Network and Distributed System Security*, 2004.

[12] Needham, R. and Schroeder, M. Using encryption for authentication in large networks of Computers, *Commun. of the ACM 21*, 12 (1978), 993-999.

[13] Kawaguchi, N., Ueda, S., Obata, N., Miyaji, R., Kaneko, S., Shigeno, H., and Okada, K. A secure logging scheme for forensic computing**.** In *Proceedings of the IEEE Workshop on Information Assurance*, 2004.

[14] Pellegrin, F. and Pellegrin C. Secure logging over a network. *Linux Journal 74es*, 10 (2000).

[15] NCSA, University of Illinois Urbana Champaign. Commands Reference, Volume 5. SyslogdDaemon. http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/IBMp690/IBM/usr/share/man/info/en_US/a_doc_lib/cmds/aixcmds5/syslogd.htm November 2008.

[16] Hall, N. Creating a secure linux logging system. SANS Institute, 2004.

[17] Jiqiang, L., Zhen, H., and Zengwei, L. Secure audit logs server to secure logs to support computer forensics in criminal investigations. In *Proceedings of the 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, October 2002.

[18] Snodgrass, R.T., Yao, S.S., and Collberg, C. Tamper detection in audit logs. In *Proceedings of the International Conference on Very Large Databases*, 2004, pp. 504 - 515.

[19] Dixon, P.D. An overview of computer forensics. *IEEE Potentials 24*, 5 (Dec. 2005), 7-10.

[20] Bishop, M. *Introduction to Computer Security*. Pearson Education, 2004.

[21] IETF Tools. Reliable Delivery for syslog draft-ietf-syslog-reliable-12.txt. 2001. http://tools.ietf.org/html/draft-ietf-syslog-reliable-12l. November 2008.

[22] Balabit.com. Syslog-ng logging system. 2008. www.balabit.com/network-security/syslog-ng/. November 2008.

[23] Kelsey, J. M. Syslog-sign and syslog-auth. In *Proceedings of the 49th Internet Engineering Task Force*, 2000.

[24] Employees.org. Syslog-Sign Protocol draft-ietf-syslog-sign-12.txt. 2003. http://www.employees.org/~lonvick/draft-ietf-syslog-sign-12.html. November 2008.

[25] Download Junction. Get/read Hard Disk Serial Number. 2008 http://www.downloadjunction.com/product/software/116855/index.html. November 2008.

[26] Sedgwick, R. *ALGORITHMS,* 2nd ed. Addison-Wesley, 1988.

[27] Serge Vakulenko. Hash function efficiency. 2008. http://www.vak.ru/doku.php/proj/hash/efficiency-en#test_1 November 2008.

[28] Monteiro, S. D.S. and Erbacher, R.F. An authentication and validation mechanism for analyzing syslogs forensically. ACM SIGOPS Operating Systems Review *42*, 3 (2008), 41-50.

[29] Monteiro, S.D.S. and Erbacher, R.F. Exemplifying attack identification and analysis in a novel forensically viable syslog model. In *Proceedings of the 3rd IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering,* 2008, 57-68.

[30] Casey, E. Error, uncertainty, and loss in digital evidence. *Int'l J. Digital Evidence 1*, 2 (2002), 1-45.

[31] Kurzban, S. Authentication of computer generated evidence in United States federal courts. *The J. of Law and Technology* (1995), http://www.ipmall.org/hosted_ resources/IDEA/35_IDEA/35-4_IDEA_437_Kurzban.pdf.