

An Authentication and Validation Mechanism for Analyzing Syslogs Forensically

Steena Dominica Steven Monteiro
Department of Computer Science, UMC 4205
Utah State University
Logan, UT 84322
steena.m@aggiemail.usu.edu

Robert F. Erbacher
Department of Computer Science, UMC 4205
Utah State University
Logan, UT 84322
Robert.Erbacher@usu.edu

Abstract

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 [4]. Currently, a patchy log on the server side and client side cannot be considered as formal authentication of a wrong doer [5]. This paper 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 a 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.

Categories and Subject Descriptors

H.2.0 [Databases]: Security, Integrity, and Protection; K.5.1 [Legal Aspects of computing]: Hardware/Software Protection; K.6.5 [Management of Computing and Information Systems]: Security and Protection – Authentication, Unauthorized access.

General Terms

Reliability, Security, Legal Aspects, Verification.

Keywords

Forensic Validity, System Log Files, Authentication and Validation, Model.

1 INTRODUCTION

This research aims to provide a mechanism that will validate and authenticate syslogs for computer forensic analysis. Syslogs are often smoking guns [24] in an organization where a computer or network attack has occurred due to the immense amount of information they contain. 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 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 [17]. 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 [1] [19] [22]. 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 [14] 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 [15].

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 detection of modifications and deletions.

2 BACKGROUND

Every computer-based activity on a system typically leaves an electronic trace [24]. 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 paper, we propose a new electronic trace by using a modification of the Needham Schroeder protocol [16]. 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.

2.1 Previous Work

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. The paper by Waters et al. [22] presents 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 [19].

Ayrapetov et al. in their paper [1] 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 [11] 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 [18] makes use of syslogd [25] 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 [9] 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. The research presented by Jiqiang et al. [10] presents 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 in the paper 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. Their 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. [21] 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 [21]. 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.

2.2 Weaknesses of the Syslog Protocol

The weakness of the syslog protocol [15] lies in the fact that it uses 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.

2.2.1 *Changing 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 generate 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.

2.2.2 *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.

2.2.3 *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 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.

2.3 Forensics Requirements

Dixon et al. [8] identified the primary characteristics that computer forensic evidence entails. This includes:

- 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.

Matt Bishop [3] specified 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.

2.3.1 *Preservation*

“Data should be obtained from forensically viable and secure systems or entities.” [8] 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.

2.3.2 *Identification*

“There must be a method or technique in place to authenticate the collected evidence.” [8] 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.

2.3.3 *Extraction*

“The extraction of digital evidence from affected systems should be carried out to preserve the evidence and not have changes in it during the collection and extraction process.” [8] 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.

2.3.4 *Documentation*

“The chain of custody and the entire procedure of evidence collection and extraction should be documented appropriately to serve as additional evidence in court.” [8]

2.3.5 *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

1. “Discovery, recovery, preservation and control of electronic data or documents.” [24]
2. “The analysis, verification, and presentation of e-evidence in court or investigations.” [24]

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

2.3.6 Confidentiality

Confidentiality refers to the concealing of a resource or a system from entities that “do not have the need to know.” [3] 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

2.3.7 Integrity

Integrity (with respect to a secure syslog) 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.

2.3.8 Availability

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

2.4 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. Two existing systems dealing with secure syslog facilities offered the greatest capability on which to build, including syslog-sign and syslog-auth. Other variants such as syslog_reliable [28] and syslog_ng [29] 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.

2.4.1 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. [12].

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 [26]. 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.

2.4.2 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 is hence better than Syslog-sign [12].

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

3 MODEL OVERVIEW

Figure 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,

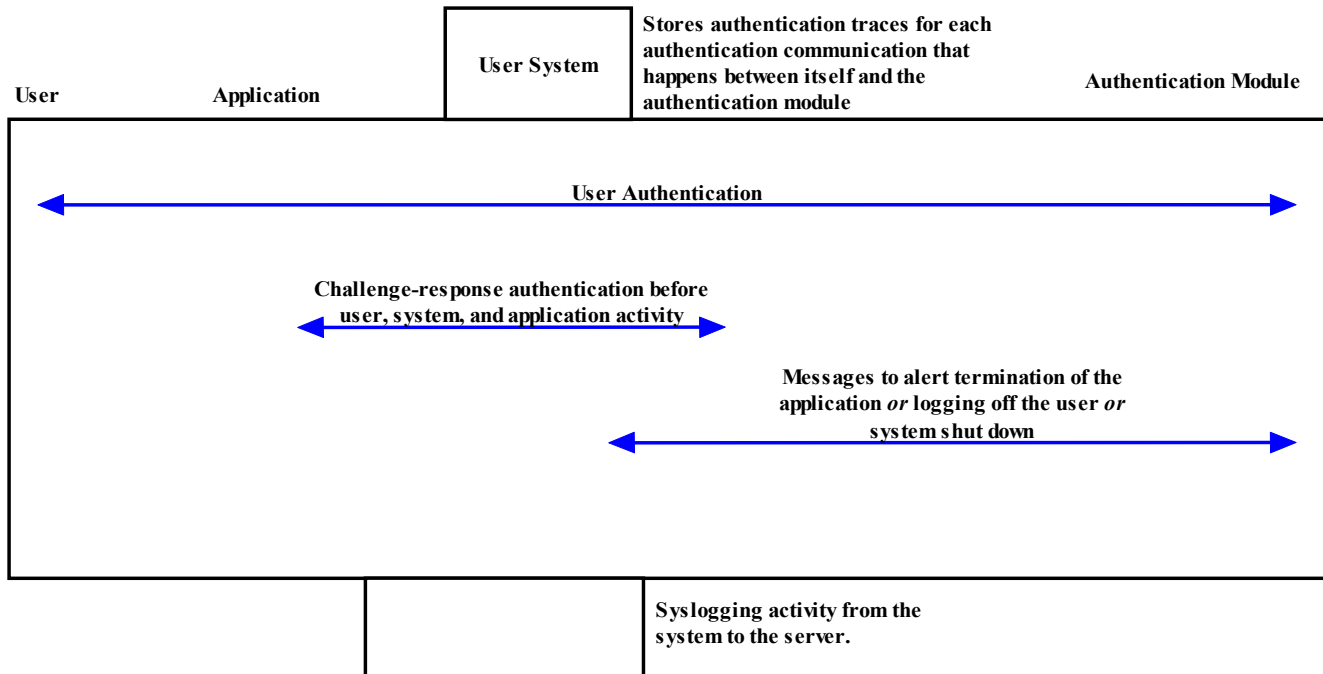


Figure 1: Model overview.

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 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.

4 THE PROPOSED METHOD

Our proposed protocol, exemplified in figure 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

4.1 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.

4.2 Phase 2: System Connection Establishment

{systemprint, userprint, randomNumber}_{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

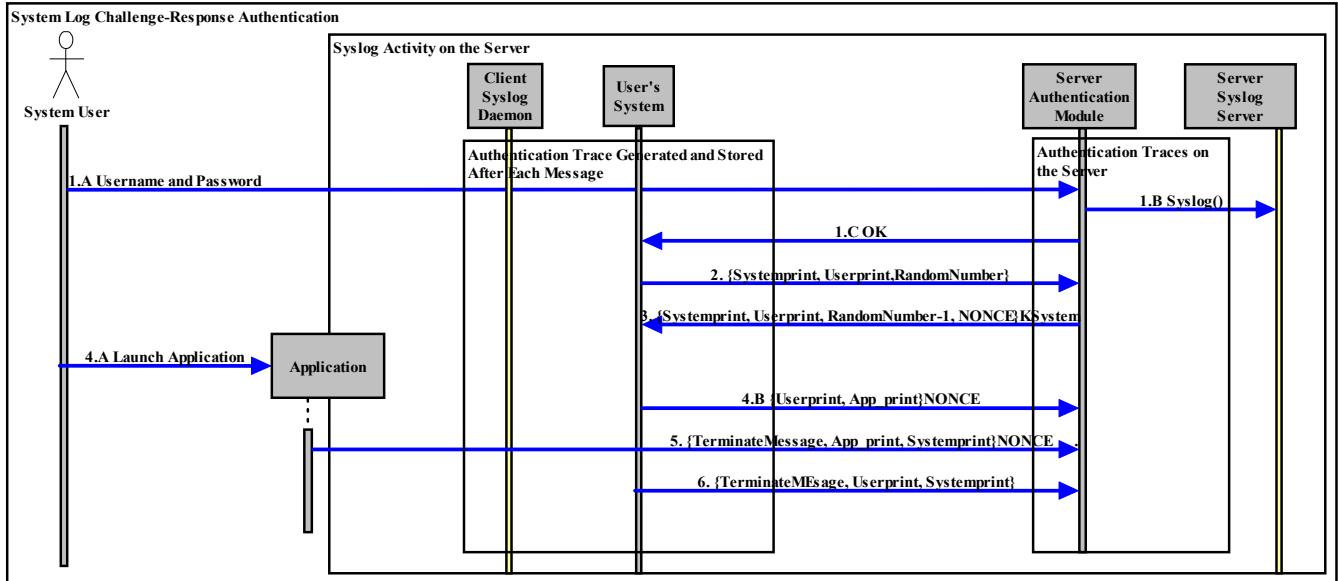


Figure 2: Sequence diagram of the proposed system. The entities are represented along the x-axis and time is represented along the y-axis.

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.

4.3 Phase 3: System Connection Establishment Response

{systemprint, userprint, randomNumber-1, NONCE}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 user.

4.4 Phase 4: Application Event Entry Generation

{userprint, app_print}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 will prevent 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.

4.5 Phase 5: Applications Termination

{terminatemessage, app_print, system_print}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.

4.6 Phase 6: System Connection Termination

{terminatemessage, user_print, system_print}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 for an intruder to compromise the log reporting facility.

5 FINGERPRINTS

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.

5.1 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 then 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.

5.2 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.

5.3 System Fingerprints

System fingerprints are often used by operating systems manufacturers to register the system that the operating system was installed on 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 will 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 [30]. This identifier should prove to be particularly effective as a system fingerprint.

5.4 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 will typically be a message along with the prints and the timestamp of the event.

The RS algorithm, which is a general-purpose hashing algorithm

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

```
C:\Program Files\Internet Explorer\IEXPLORE.exe terminated at 2007-12-30 06:51:13 with ID
4384844220178160764 with fingerprint 223266966
```

(a)

```
Incorrect login with username: ghost occurred at 2007-12-28 23:01:16 with userID
2221344687639655740 with userprint 238806054
```

(b)

Figure 3: 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.

developed by Robert Sedgwick [20] is used to generate hashes, i.e., fingerprints. A test carried out [27] shows that the RS algorithm had very few collisions when tested on a huge string data set. Example valid and invalid authentication traces are shown in figure 3.

5.5 NONCE

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 will prevent 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 the associated authentication traces and fingerprints.

6 FORENSIC VIABILITY

Previous research has dealt with using digital evidence in a court of law as documented in [5]. Forensically viable log files as defined in [6] 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 [6] 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* [23]. 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.

The research in [5] 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 1.

According to this table, it is clear that the evidence presented by the syslogs, which are collected and generated by our approach, fall into the C5 level; given the authentication and validation capabilities integrated into the model. Whereas 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 [6].

Table 1: Mapping the certainty levels defined in [5] to syslog files.

Level	Level Confidence	Relationship to Syslogs
C0	Erroneous/Incorrect	Programmatic errors while coding the syslog/syslogd protocol. [15]. 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.

7 PROTOCOL RESILIENCE

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 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.

7.2 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.

7.3 Phase 4: Application Event Entry Generation

The nonce used to encrypt this message is generated by the server and 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.

7.4 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.

7.5 Phase 6: System Connection Termination

Man in the middle attacks will 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.

7.6 System Log Truncation

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 will succeed 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 will not eliminate the fact that a certain event had occurred since the logs on the authentication server will still have evidence of every communication that has occurred.

7.7 Denial of Service Attacks

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 will have 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 will 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.

7.8 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 will 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.

7.9 Hostile Applications

Installation of hostile applications on a system is a common technique that intruders use to obtain information about the network and monitoring activities covertly. In this scenario, the goal of the hostile application will be to imitate an application that is already running on the system and to trick the system into believing that it is indeed the valid application. The hostile application will attempt to inject false events or block transmittal of valid messages. In our model, in order to be launched, the hostile application will need to authenticate with the server. This will not be possible since the application will need to have a valid application print authorized by the server.

The strength of our proposed handshaking mechanism lies in the fact that it uses nonces and random numbers. This makes it resistant to nonce number prediction attacks, which is comparable to the sequence prediction attacks observed in TCP handshakes [2].

7.10 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 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.

8 CONCLUSIONS

The proposed method aims to provide 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 and the need for legal admissibility. The fingerprints assigned to every entity involved in system log generation will enable the validation of these entities. More importantly, since digital evidence is treated in the same way as documentary evidence [13], a means to authenticate and verify its authenticity is needed. The proposed method aims to provide 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 could easily be attacked in court and invalidated.

With the proposed method, if a malicious insider carries out suspicious activity, this activity can be traced back to the offender. Their 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 be able to trace back to insiders. 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.

9 REFERENCES

- [1] Aryapetov, D., Ganapathi, A., and Leung, L., "Improving the Protection of Logging Systems," Technical Report, UC Berkeley, 2002.
- [2] Bellare, S.M., "Security Problems in the Tcp/Ip Protocol Suite," *ACM SIGCOMM Computer Communication Review* Vol. 19, No. 2, 1989, pp. 32-48.
- [3] Bishop, M., *Introduction to Computer Security*, Pearson Education, October 2004.
- [4] Brezinski, D. and Killalea, T., "Guidelines for Evidence Collection and Archiving," RFC 3227, 2002.
- [5] Casey, E., "Error, Uncertainty, and Loss in Digital Evidence," *International Journal of Digital Evidence*, Vol. 1, No. 2, 2002, pp. 1-45.
- [6] Ceresini, T., "Maintaining the Forensic Viability of Logfiles," SANS Institute, SANS Institute, no. As part of GIAC practical repository, 2003.
- [7] Dinant, J. M., "The long way from electronic traces to electronic evidence," *International Review of Law, Computers & Technology*, Vol. 18, No. 2, July 2004, pp. 173-183.
- [8] Dixon, P.D., "An Overview of Computer Forensics," *IEEE Potentials*, Vol. 24, No. 5, Dec. 2005, pp. 7-10.
- [9] Hall, N., "Creating a Secure Linux Logging System." SANS Institute (2004): As part of GIAC practical repository.
- [10] L. Jiqiang, H. Zhen, and L. Zengwei, "Secure Audit Logs Server to Secure Logs to Support Computer Forensics in Criminal Investigations," Proceedings of the 2002 IEEE Region 10 Conference on *Computers, Communications, Control and Power Engineering*, October 2002.
- [11] Kawaguchi, N., Ueda, S., Obata, N., Miyaji, R., Kaneko, S., Shigeno, H., Okada, K., "A secure logging scheme for Forensic Computing," Proceedings of the *IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, 2004, pp. 386- 393.
- [12] Kelsey, J. M., "Syslog-Sign and Syslog-Auth," Proceedings of the *Forty-ninth Internet Engineering Task Force*, San Diego, CA, USA 2000.
- [13] Kurzban, S., "Authentication of Computer Generated Evidence in United States Federal Courts." *The Journal of Law and Technology*, 1995.
- [14] Lonvick, C., "RFC 3164-The BSD Syslog Protocol," Cisco Systems, August 2001.
- [15] Nawyn, K. E., "A Security Analysis of System Event Logging with Syslog." SANS Institute, no. As part of the Information Security Reading Room. (2003).
- [16] Needham, R. and Schroeder, M., "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, Vol. 21, No. 12, 1978, pp 993-999.
- [17] Peisert, S., "Forensics for Network Administrators," *USENIX*, 2005, pp. 34-42.
- [18] Pellegrin, F. Pellegrin and C., "Secure Logging over a Network," *Linux Journal*, Vol. 74es, No. 10, 2000.
- [19] Schneier, B. and Kelsey, J., "Secure Audit Logs to Support Computer Forensics," *ACM Transactions on Information and System Security*, Vol. 2, No. 2, 1999, pp. 159 - 176.
- [20] Robert Sedgwick, *ALGORITHMS*, Second Edition, Addison-Wesley Publishing, Company, Inc., 1988.
- [21] Richard T. Snodgrass, Shilong Stanley Yao and Christian Collberg, "Tamper Detection in Audit Logs," In *Proceedings of the International Conference on Very Large Databases*, Toronto, Canada, 2004, pp. 504 - 515.
- [22] B. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an Encrypted and Searchable Audit Log," *Eleventh Annual Symposium on Network and Distributed System Security (NDSS'04)*, San Diego, California, 2004.
- [23] Veritect Inc. "What Lawyers and Managers Should Know About Computer Forensics," January 2001, URL: http://www.veritect.com/about_veritect/comp_forensics.pdf.
- [24] Volovino, L., "Electronic Evidence and Computer Forensics," *Communications of the Association for Information Systems*, Vol. 12, 2003, pp. 457-468.
- [25] <http://www.scit.wlv.ac.uk/cgi-bin/mansec?1M+syslogd>
- [26] <http://www.ietf.org/internet-drafts/draft-ietf-syslog-sign-1.txt>
- [27] http://www.vak.ru/doku.php/proj/hash/efficiency-en#test_1
- [28] <http://www.ietf.org/internet-drafts/draft-ietf-syslog-sign-21.txt>
- [29] www.balabit.com/network-security/syslog-ng/
- [30] <http://www.downloadjunction.com/product/software/116855/index.html>