UDC 004.45

SECURE ACTIVITIES MONITORING AND AUDITING IN THE CLOUD

Lyudmila Zharova[1], Ivica Stankovic[2], Radomir Mihajlovic[3]
[1]NYIT, New York, NY, USA, lzarova@nyit.edu
[2]Faculty of Business Studies and Law, Univ. UNION-Nikola Tesla, Belgrade Serbia, ivica.stankovic@fpsp.edu.rs
[3]NYIT, New York, NY, USA, rmihajlo@nyit.edu

Abstract: *Faced with the current trend of enterprise computing infrastructure being relocated from physical data centers into the cloud, systems and applications security administrators as well as developers are forced to dedicate particular attention to new set of security problems. When making decision about what sort of cloud service to use, the public cloud appears to be the most cost effective. However, to use the public cloud service decision makers have to trust the cloud service provider, who has complete access to the file storage infrastructure and all client files including security relevant system and application log files. In this paper we make a parallel between the financial and computing system event monitoring, we present key clarifying definitions and propose a encryption based log file data privacy solution by selectively protecting only privacy-critical log records. Our solutions do not impose any demands on the existing systems solutions or current systems logging infrastructure maintenance On the contrary, the only conditions are imposed on developers of cloud privacy-sensitive applications and designers of log parsers and analysis tools. In summary, our solutions move system log privacy problems in the cloud into the application layer.*

Keywords: *Cloud, security, system monitoring, auditing, accountability, forensics, XML, digital signature, encryption.*

1. INTRODUCTION

After the hardware platform, the operating system and applications have been well hardened and after all protective programs have been installed, configured, loaded and activated, monitoring system activities and periodic auditing of all collected auditing data of security importance remains as important systems security administrators duty.

Maintaining a secure operating system requires administrator's vigilance, because the original security configuration for any system tends to fail over time. Security and performance auditing, i.e., examination of the system readiness is an essential component of the operational and security maintenance. Probably the most valuable tools available to a security administrator are tools that enable an administrator to continuously monitor and record specific computing events. The ability to record computing events visible from the systems point of view is valuable in particular when potential legal action may have to be taken against an identified system attacker.

Probably the most difficult types of cases to prosecute in the courts of law today are those cases that involve computer security breaches or system intrusions. Prosecution is made difficult by the fact that computer technology and its practical use are far ahead of the common prosecutor's expertise and laws that are not sophisticated enough. Many states and countries take different positions on what constitutes a crime in the electronic domain. Even if there were definitive criteria that could be universally applied, prosecutors would still have the difficult task of providing proper evidence. In such situations, the more evidence the better.

Overwhelming computing systems complexity, the nature of computer related crimes and the lack of proper evidence makes it obvious that absolute system protection is virtually impossible. With this in mind we must recognize the importance of the consistent system monitoring and auditing. For instance, computing system administrators need to monitor system activities and must be able to answer at any time the following questions:

- What do users do while using the system to run their applications?
- Which user's programs have access rights to which computing resources?
- What are normal systems events?
- What is normal computing load?
- What events need attention and what kind of attention?
- and so on.

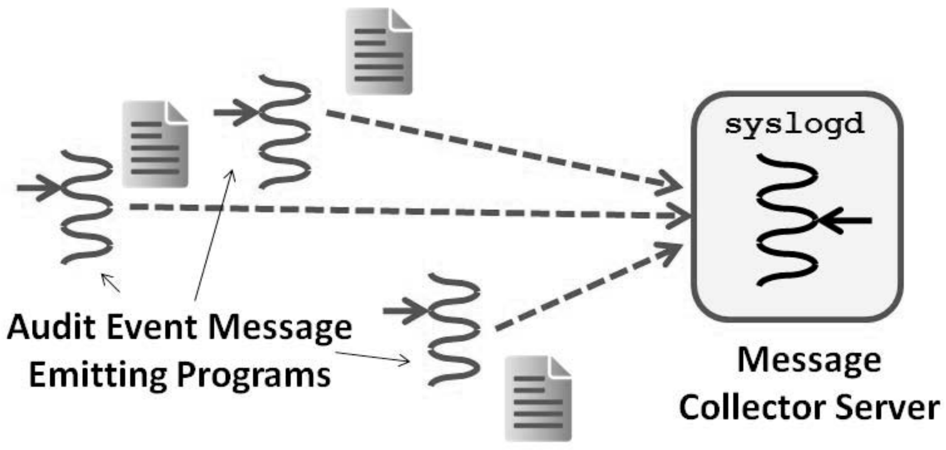


Figure 1:Systems and application programs emit systems relevant event descriptions which are collected, classified and recorded as logs by a log collector such as the `syslogd` server, [5] [6].

The ability to answer these questions implies continuous system tracking and monitoring of the activities of individual user programs, better known as application programs.

Each operating system is delivered with some sort of system monitoring and auditing tool set. One such a de-facto standard tool set is the UNIX/Linux syslog computing event logging facility [1], [2]. In response to the numerous security deficiencies of the syslog facility a number of more secure non standard replacements have been proposed [3], [4].

While designers of safer and more efficient replacements of syslog facility are focused on the fact that applications have to remain unaffected and that applications have to follow the same API to emit messages, our approach is to modify API by adding a layer of security enhancing software in order to preserve the privacy of logging messages in environments such as the public cloud or open networks.

2. SYSTEM EVENTS TRACKING

Computing events visible in the operating environment can be considered to be systems events. Dedicated operating system event tracking facilities collect various process generated messages (See Figure 1) and produce preliminary reports of events that took place during the execution of systems or application programs. These preliminary reports, are made up of chronologically ordered, time-stamped event descriptions, are known as systems journals or logs. We refer to such events as systems audit events. For instance, any attempt to use a disabled user account could be an instance event to be emitted by the systems access control software and recorded by the event tracking message collector server, such as the `syslogd` program.

There are several ways to define an event. For example, a clipping level or threshold may be set to generate an event which is to be reported and recorded, e.g., when more than three failed logon attempts in a short period of time could represent an audit event which has to be tracked. The declaration that a given system audit event took place is based on a predefined event criteria of an if-then kind.

Events are tracked by the event time and description recording in order to inspect the recorded data at some later time, at the so called audit time. Every event that results in some negative consequences has to be associated with the user whose action directly or indirectly caused the event. In most operating environments every event recorded should have an owner, i.e., a person accountable for the event.

Definition 1: Accountability is characteristic or an attribute of a subject that is conceptually rooted in ethics and law with several implicated characteristics (meanings) like

- answerability,
- responsibility,
- blameworthiness, and
- liability

Systems journals or logs are raw chronological reports with time stamped event descriptions so that administrator may walk back through the sequence of logged events in order to discover the cause of some operational problem or some intrusion event. Sequential inspecting of logged records is a fundamental system auditing procedure, the so called audit trail. Auditing is a common activity in business, finance and accounting. In all cases of auditing the first documents to be audited are accounting journals which are time stamped financial event records equivalent to system logs. Similar to financial records and journals auditing, we have systems and log files auditing for two possible reasons:

- Operational and management improvement, or
- Security and safety reasons.

Besides the accounting audit trail it is possible to perform operating systems security audit trail too. Security audit trail may be defined in the following manner:

Definition 2: A security audit trail is a chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments (relevant objects) and activities of subjects surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to final results.

An accountability subsystem is a collection of systems components used to associate a subject (e.g., some application user) with its past actions performed upon some computing resources, with past associations time stamped and documented. Accountability documents are known as journals, log records or audit trails.

From the security point of view logs are important for:

- Detecting security violations (detecting intrusion events),
- Re-creating security incidents, and
- Forensic security violation investigations, (production of evidence).

From the operational administration and management point of view audit event tracking logs are used for:

- System performance improvements,
- System administration improvements, and
- Production of system accounting management reports.

Accountability record management requires that audit event tracking systems be efficient and secure. However, if administrators do nor review the logs in a timely manner or if logs are not produced and maintained in a secure and consistent manner, the logs may not be admissible as evidence in a court of law and may not be of any use for and operational needs. In our work we focus on the security of log messages that must cover two fundamental requirements:

- Message integrity (message should be modification protected), and
- Message privacy (message should be readable only by the intended final user).

Both of the above log message protection requirements are essential when log messages become exposed to the cloud providing service or open network users.

For better log management a typical log message is sent to a central logging facility (See Figure 2). According to the IETF specifications [5] systems audit event tracking involves log message transmission from the message source program and transmission between log collection servers that may be one of three operating entities:

- Devices, (original receivers of messages from systems programs and application processes in the given host),
- Relays (log data collectors and forwarding servers), and

- Collectors (final log data collecting destination).

In our work we focus on the very first entity, i.e., on the application program which is the source of the log-message and message security until it is used for log analysis. Log message content must be protected from the source to the destination.

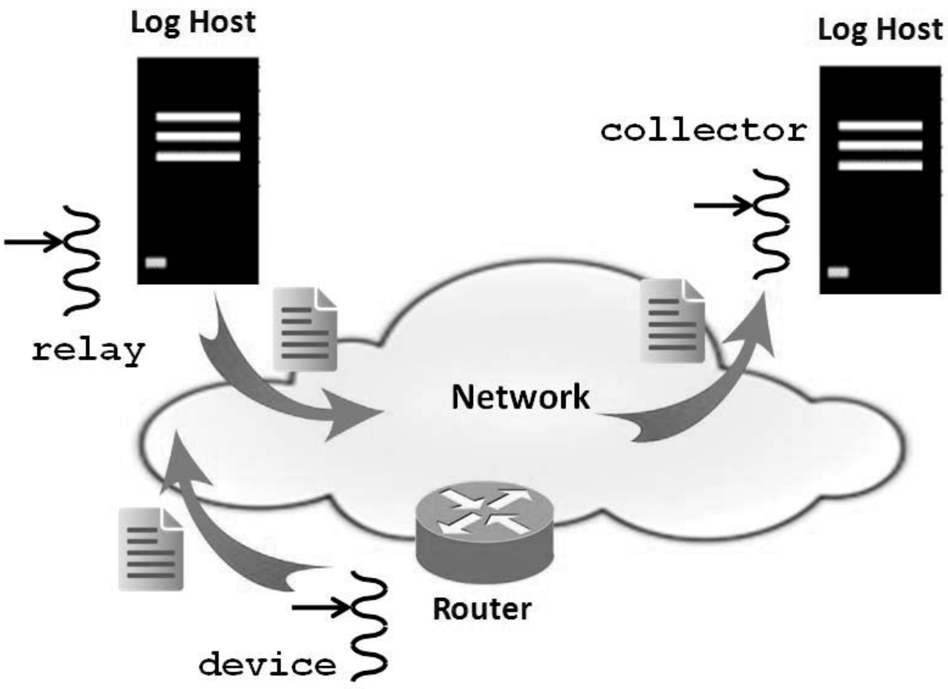


Figure 2: Central logging of syslog messages.

Monitoring the log files on multiple hosts can become administration unfriendly as the number of networked hosts increases. Configuring a centralized log message collection on a central logging server can:

- Reduce the administrative burden of log data management, and
- Enable faster response to some adverse events.

As Figure 2 illustrates, various network traffic processing devices, including routers, switches, firewalls, virtual private network concentrators, and so on, have operating systems with logging facility capable of sending log messages to other systems logging facilities. Most of the secure log message exchange across the network are based on some sort of message encryption tunnel service such as SSH or IPsec. We take a different approach by delegating log message security problem solving to an original application program and log analysis software developers.

3. LOG MESSAGE PROTECTION

Typical log data processing server is customizable, i.e., configurable. System administrator customizes log facility server by altering certain configuration variable values which appear as some sort of instructions to the server on how to perform its service. Server programs read configuration parameter values:

- By default at load time, or
- On demand when instructed to do so, at run time.

Default reading of configuration parameters at load time can be performed using three different methods:

- Reading parameter values introduced as command line arguments when starting the server,
- Reading parameter values defined as environment or shell variables, or
- Reading parameter values found in the appropriate configuration file.

In case that the same parameter has three different values, one presented at the command line takes precedence over the shell variable equivalent, which would take precedence over the one presented in the configuration file.

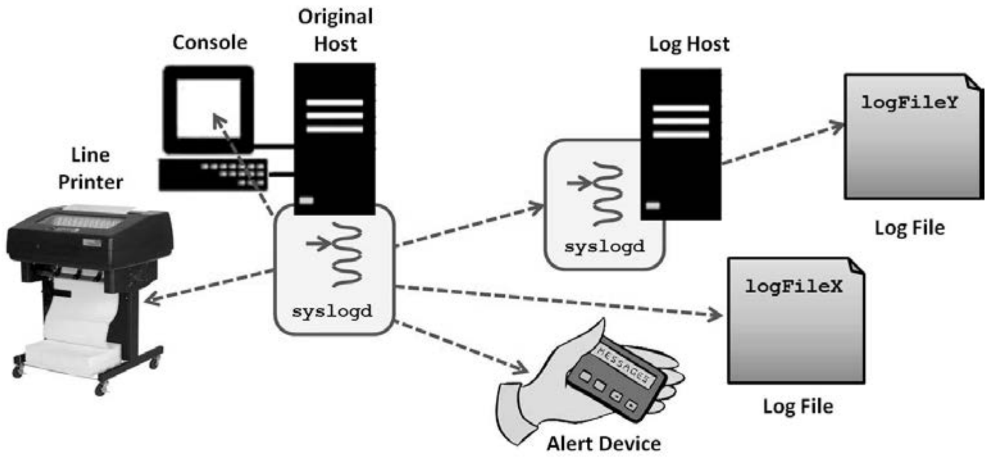
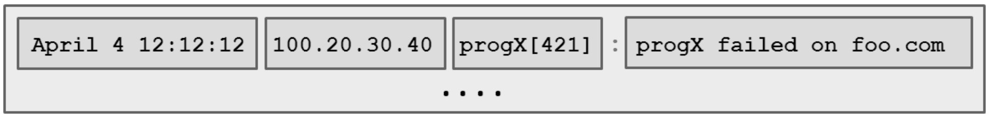


Figure 3: Possible destinations of log messages.

A system event logging facility such as one based on the `syslogd` daemon, receives log messages from various programs and routes each message to the destination specified by the configuration parameters (See Figure 3). An example of the log message or log text line is shown in Figure 4. Single line of logged text is made up of two distinct parts:

- The "header" which contains the date-and-time stamp, and
- The event identifying text string.



```
April 4 12:12:12 | 100.20.30.40 | progX[421] : progX failed on foo.com
....
```

Figure 4: An example of a log line generated by the `syslogd` logging service.

In the example shown in Figure 4, clear text string describes the logged event details which are:

- The message originating system's hostname or IP
- The message generating source type or facility, and
- The descriptive message.

Since all logged data are processed in clear, i.e., are open and readable even when some details should be guarded as private, we propose not to interfere with any existing logging facility requirements and operational details, but to have privacy sensitive application programs perform encryption of the clear message portions and to send each message digitally signed.

Major restriction found in the standard system monitoring solutions is to limit event description text string to a single line with fewer than 1024 characters. Consequently, our log record privacy solution had to use cryptographic algorithms and binary-to-text data codes that would produce as short strings of text as possible. Based on this argument, data codes like the base64 code are favored. Lines which would upon protection become longer than allowed, are not subjected to any protection.

The scope of this report does not permit detailed presentation of the cryptographic algorithms used. In what follows, only functional details are outlined. For example, let us consider small block of log lines:

```
03/03/2016 08:34:21 -> User 'Bill' logged in
03/03/2016 08:34:28 -> Navigated to google page
03/03/2016 08:34:32 -> Option A changed to B
```

A developer of the privacy sensitive application, uses system API to identify computing process (to get PID) of the program at runtime. To identify precisely the source of the log message, the acquired PID can be embedded into each log message. Furthermore, we propose XML as log message structure defining language, i.e., it is expected that the original application program generate XML formatted log lines which may appear as shown below:

```
03/03/2016 08:34:21 -> <p id=103>'Bill' logged in</p>
03/03/2016 08:34:28 -> <p id=204>Navigated to google page</p>
03/03/2016 08:34:32 -> <p id=204>Option A changed to B</p>
```

The fine grain identification of the original log line segments is additional feature that our approach offers. Parsing of the XML structured log line is simpler and more flexible.

To protect log line integrity, i.e., to prevent illegal alterations, we propose truncated digital signature which is represented by the **<s></s>** XML element:

```
03/03/2016 08:34:21 -> <p id=103><m>'Bill' logged in</m><s>5g$haa6bc3</s></p>
03/03/2016 08:34:28 -> <p id=204><m>Navigated to google page</m><s>2035a6bf78</s></p>
03/03/2016 08:34:32 -> <p id=r204><m>Option A changed to B</m><s>450b25c3e1</s></p>
```

When log line privacy is of the essence, the text portion of the log line is encrypted, (e.g., public key presented by the log analyst's certification authority may be used). In our example, with both integrity and privacy protected, the text is not readable while the time stamp is.

```
03/03/2016 08:34:21 -> <p is=103><cm>0f98+5h%m*2o84765c&!.)s2</cm><s>2035a6bf78</s></p>
03/03/2016 08:34:28 -> <p id=204><cm>s3#08b&jkr71(/0*hd5%28*v</cm><s>2035a6bf78</s></p>
03/03/2016 08:34:32 -> <p id=204><cm>a@k)n56l3#sp2(+g$/sdf20e</cm><s>450b25c3e1</s></p>
```

We have performed several application experiments using shell scripting, C, Java and PhP programming languages. In order to minimize XML overhead [7] and binary-to-text encoding overhead, we use minimal XML tag lengths and we favor data codes such as base64 code.

5. CONCLUDING REMARKS

Our work presented in this report addresses problems of secure systems operational intelligence gathering using standards such as syslog facility while focusing on technical issues of how to preserve selected log lines content privacy and integrity.

We keep in mind that system and application attackers desire to alter log lines that could possibly be used as forensic evidence. In order to prevent hiding attack tracks by altering logs we use digital signature and we hide sensitive line content by encrypting selected lines and even portions of sensitive lines.

When business case would demand privacy, all reported solutions offer encryption of the complete log file, which apparently is a coarse grain log privacy preservation method. Our approach as a fine grain method applies encryption and digital signature only on selected sensitive log lines and their distinguished portions.

REFERENCE

[1] C. Lonvick, "The BSD syslog Protocol," IETF Network Working Group, RfC-3164, August 2001. https://tools.ietf.org/pdf/rfc3164.pdf

[2] R. Gerhards, "The Syslog Protocol," IETF Network Working Group , RfC-5424, Standards Track, March 2009. https://tools.ietf.org/html/rfc5424

[3] Peter Matulis, "Centralised logging with rsyslog," Technical White Paper Canonical, September 2009. http://insights.ubuntu.com/wp-content/uploads/Whitepaper-CentralisedLogging-v11.pdf

[4] "The syslog-ng Open Source Edition 3.7 Administrator Guide," BalaBit S.a.r.l., April 04, 2016. https://www.balabit.com/documents/syslog-ng-ose-3.7-guides/en/syslog-ng-ose-guide-admin/pdf/syslog-ng-ose-guide-admin.pdf

[5] D. New, M. Rose, "Reliable Delivery for syslog," IETF, RfC-3195 Standards Track, November 2001, https://tools.ietf.org/pdf/rfc3195.pdf

[6] Radomir A. Mihajlovic, Aleksandar R. Mihajlovic, "Operating Systems Security; The First Cut," Soft Electronics, ISBN-978-194327525-0, New York, May 2015.

[7] Barbara Carminati, Elena Ferrari, Elisa Bertino, "Securing XML Data in Third-Party Distribution Systems," CIKM '05 Proceedings of the 14th ACM international conference on Information and knowledge management pp.99-106. https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/bibtex_archive/2005-117-report.pdf