Implementing Forensic Readiness Using Performance Monitoring Tools

Franscois Van Staden, Hein Venter

Chapter 18

IMPLEMENTING FORENSIC READINESS USING PERFORMANCE MONITORING TOOLS

Franscois van Staden and Hein Venter

Abstract This paper proposes the use of monitoring tools to record data in support of digital forensic investigations. The collection of live system data requires integrity checks and data validation to be performed as the data is collected and stored. Combining system monitoring and digital forensic functionality in a single system reduces the cost and complexity of administration and maintenance, while enhancing forensic readiness.

Keywords: Digital forensic readiness, performance monitoring tool

1. Introduction

Digital forensic investigators frequently have to sift through large data sets such as log files to find evidence of malicious activities. However, due to the storage constraints imposed on live systems, log files are often rotated (i.e., old data is overwritten with new data) in an attempt to save space. The process of log file rotation can cause valuable data to be lost. Therefore, when an incident is detected, the production systems need to be stopped until the relevant data has been collected by the investigator.

Performance monitoring systems are commonly used to collect data about system operations (e.g., CPU, memory and hard drive utilization). Custom probes and reporting facilities can be instituted to extend traditional performance monitoring. In particular, probes can be created to identify specific data of interest in log files or to collect audit data stored in an application database. The data collected by the probes

can be stored in a central monitoring server and used to create various reports.

This paper discusses the use of monitoring tools to record data in support of digital forensic investigations. The collection of live system data requires integrity checks and data validation to be performed as the data is collected and stored. Combining system monitoring and digital forensic functionality in a single system reduces the cost and complexity of administration and maintenance, while enhancing forensic readiness.

2. Background

This section provides an overview of performance monitoring, digital forensics and digital forensic readiness.

2.1 Performance Monitoring

Performance monitoring tools are designed to collect data about software systems in order to report on performance, uptime and availability. Live data about the monitored systems is used to detect system problems and pinpoint their source. Several performance monitoring tools employ the Simple Network Monitoring Protocol (SNMP) [1]. SNMP provides a means for connecting to and collecting data from servers, firewalls and network devices, mainly for the purpose of performance monitoring.

Data used by performance monitoring tools can be categorized as live data, historical data or custom data. Live data is collected during the latest completed collection cycle and provides information about the current system state. After a set period of time, live data is reformatted and moved to a historical data archive. This historical data provides information about system performance, uptime and availability.

Custom data may also be collected by performance monitoring tools, but the data is neither used to display the current system state nor is it converted to historical data. Instead, it is typically stored in a read-only format and presented in custom reports. Performance monitoring tools are normally unable to interpret custom data. Descriptors can be created to enable custom data to be used by performance monitoring tools, but then the data is no longer considered to be custom data.

2.2 Digital Forensic Readiness

Digital forensic science is a relatively new field of study that has evolved from forensic science. The Oxford Dictionary [3] defines digital forensic science as the systematic gathering of information about electronic devices that can be used in a court of law. Digital forensic

science is more popularly referred to as digital forensics or computer forensics.

Palmer [4] defines digital forensics as "the use of scientifically derived and proven methods towards the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence from digital sources for the purpose of facilitation or furthering the reconstruction of events." Note that Palmer's definition describes the process of digital forensics whereas the Oxford Dictionary defines the science of digital forensics. Another definition of the process of digital forensics is provided by Kohn, *et al.* [2], who define a digital forensic process having "an outcome that is acceptable by law."

Rowlingson [5] defines digital forensic readiness in terms of two objectives. The first objective is to maximize the ability of the environment to collect digital forensic data and the second is to minimize the cost of a forensic investigation. In order to prepare an environment to be forensically ready, a mechanism is needed to preserve, collect and validate the data contained in the environment. This data can then be used as part of a digital forensic investigation.

2.3 Protocols

The Lightweight Directory Access Protocol (LDAP) [7] specifies a mechanism for accessing distributed directory services. Our work engages the OpenLDAP, an open source implementation of the LDAP specification, which is used as a single point of authentication for users who access network resources.

Secure Shell (SSH) [6] is often used to implement secure communications over insecure networks. The performance monitoring tool described in this paper uses SSH to establish communications with its probes. This ensures the integrity of the data sent by the probes to the performance monitoring tool.

3. Log File Sniffing

This section describes how the performance monitoring tool is used to collect data from log files in order to enhance forensic readiness. The log files are located in various locations on multiple servers in a network.

3.1 Test Environment

The performance monitoring tool was deployed in an environment where users log into a web system comprising several web applications. The web applications, which were developed in Java, use the same session information to grant users access to services. The Java application

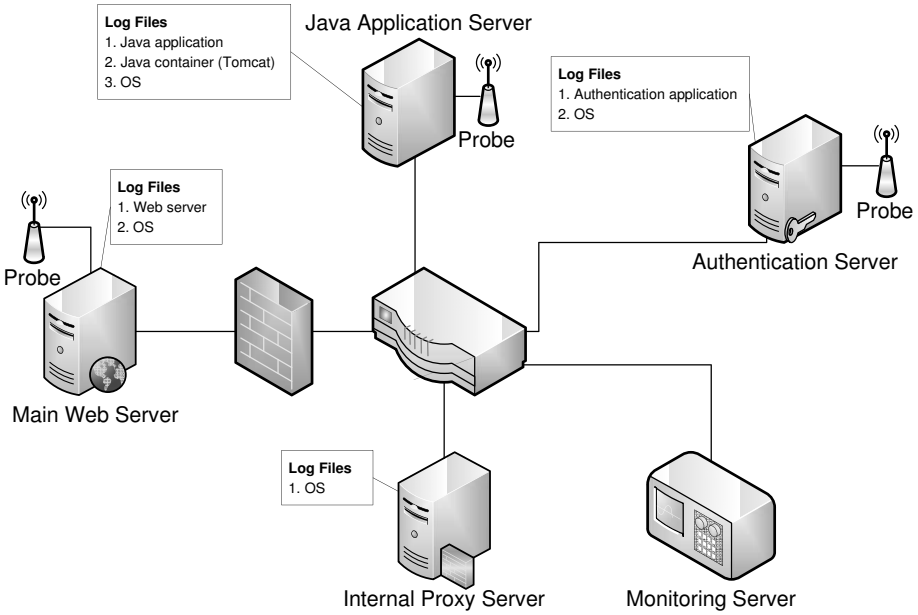

Figure 1. Test environment.

server uses Apache Tomcat application containers to manage Java applications. Users connect to the main web server to interact with the access-controlled Java applications. The main web server creates connections to the Java application server using a proxy server to hide the internal URLs of Java web applications.

Figure 1 presents the test environment, which comprises five servers: (i) main web server; (ii) Java application server; (iii) authentication server; (iv) internal proxy server; and (v) monitoring server. Authentication is implemented using OpenLDAP. An application running on the Java application server performs authentication lookup and session management when users log into any of the Java applications.

Probes are installed on the main web server, the Java application server and the authentication server. A probe is simply a device that collects data for the monitoring server. The probes are set up to sniff log files on the three servers for user login activity data and user session data. The web server log files contain data about user web resource requests. The Java application server log files contain data about user sessions and application access and authentication requests. The authentication server log files contain data about the outcomes of authentication requests.

During a user-initiated login process, a connection is established from the main web server to the Java application server through the firewall and internal proxy server. The main login page is displayed in the user's browser. When the authentication application successfully authenticates the user using OpenLDAP, the authentication application generates a session object that is saved by the authentication application and sent to the user's browser.

System log files can contain data about system state, actions performed and errors encountered. Web applications that make use of containers such as Apache Tomcat and WebLogic have additional logs that are maintained by the containers. Application log files and container log files may reside in various locations on the server. Documenting the locations of the various log files and the data they contain is helpful in forensic investigations; otherwise, assistance from a systems administrator may be needed to obtain the required information.

If systems administrators are not available and no documentation exists, investigators could spend considerable time locating and scanning the log files themselves. Setting up probes in advance to collect the log file data significantly reduces the data collection effort on the part of forensic investigators.

3.2 Data Collection

This section describes the probes used to collect data from the various log files along with the details of their operation.

3.2.1 Custom Probes. Probes can be positioned at various locations to monitor system performance, database data and log files. Probes normally do not perform any processing, but access "data points" and periodically read their values. A data point can be a system value (standard data point) or a software generated data point (extended data point). Standard data points such as CPU usage, memory usage and drive space usage are collected by the operating system. Extended data points from log files and databases are collected by applications.

A data value captured by a probe is sent to a listening agent. The listening agent is a software application installed on the monitored resource to allow for SSH communications between the performance monitoring tool and the monitored resource. Communications between the listening agent and the performance monitoring tool are established periodically, according to the timed events set when configuring the performance monitoring tool. A timed event set to five minutes means that the performance monitoring tool communicates with the listening agent every five minutes.

Figure 1 shows three probes installed as part of a performance monitoring strategy. Each probe is configured to provide specific information (CPU usage, memory usage and disk space usage) about the server on which it is installed. The probe on the main web server collects data from extended data points about the web server, including the number of requests per second, the response time per request, and the average response time of each web page. The probe on the Java application server collects data from extended data points about the Java applications and the Apache Tomcat container. This data is used to profile the Java applications in order to perform code optimizations and scale the Java environment. Data about the authentication process, such as the number of logins per minute and authentication system performance, is collected by the authentication server probe from an extended data point.

3.2.2 Log File Probes. The probes access the various log files using log file sniffing applications. Each log file has its own sniffing application. When a probe is polled for data, it queries the relevant log file sniffing application. Each application records the last line read from its log file; each subsequent read proceeds from the first line after the last read point to the end of the file. Each line is verified on the basis of the date-time stamp, log file name and server name.

The probes receive the lines as a single string, which is sent to the performance monitoring agent. The listening agent returns the received string to the performance monitoring server. As stated previously, the listening agent connects to the monitoring server using an SSH connection. This ensures that the data is not tampered with during transmission. The monitoring server stores the validated log file information in a read-only database table. This table only allows data to be stored using the SQL INSERT statement and data to be read using the SQL SELECT statement. Note that the data in the table cannot be edited using the SQL UPDATE statement or deleted using the SQL DELETE statement.

4. Using the Collected Data

Although the data is stored in a read-only database table on the monitoring server, the performance monitoring tool is unable to analyze the data. This is because the data dictionary of the performance monitoring tool was not augmented to express the semantics of the collected data. In fact, augmenting the data dictionary was deemed to be outside the scope of the research because the objective was to use the performance monitoring tool "as is" to implement forensic readiness.

Table 1. Sample application log file data.

30-Jul-2011 11:00:00; INFO; s98278178; NULL; Authentication request received
30-Jul-2011 11:00:01; INFO; s98278178; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz; Authentication request successful
30-Jul-2011 12:15:29; INFO; s12345678; NULL; Authentication request received
30-Jul-2011 12:15:29; INFO; s12345678; NULL; Authentication request failed

4.1 Collected Data

Web server requests and responses are stored in the web server log file of the main web server. Each line in the log file contains: date-time stamp, requesting address, requested resource, HTTP request type (GET/POST) and request outcome.

The authentication application log file stores user authentication and session management data. Every authentication attempt is stored in the log file. Each line in the authentication application log file contains: data-time stamp, log level, username, session ID and authentication process outcome.

Table 1 shows four lines from the application log file. A session ID is issued according to the username and date-time stamp when an authentication request is successful. The session ID is used to generate a session object that is referenced by all the web applications to verify the authentication status of a user. The session object is destroyed when the user session expires. When a session ID is not stored in the log file, the implication is that the request was made for a new authentication session (first line in Table 1). When a session ID is stored, the authentication request has succeeded (second line). The last line in Table 1 has a NULL session ID, which indicates that the authentication request has failed. Note that an authentication request can also be a request to change a user password.

Authentication requests are sent from the authentication application to the authentication server and are logged in the authentication server log file. Table 2 shows the data contained in the authentication server log file. Each line in the log file contains: date-time stamp, username, session object ID and the authentication request outcome.

Table 2. Sample authentication server log file data.

30-Jul-2011 11:00:01; INFO; s98278178; Authentication successful; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz
30-Jul-2011 11:04:00; INFO; s98278178; Validation successful; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz
30-Jul-2011 12:15:29; INFO; s12345678; Failed

Session validation occurs when a user moves from one web application to another. If the outcome of a session validation request is negative, then it normally means that the user does not have permission to access the requested application. Line 2 in Table 2 shows a session validation request.

4.2 Investigating a Brute Force Attack

This section discusses an experiment that was conducted to detect evidence of a brute force attack and attempt to determine its origin. After the log file probes were positioned, a mock brute force attack was launched against a test account. The attack was initiated from a random computer within the organization's network; the investigator had no knowledge about the attack.

Evidence of a possible brute force attack was discovered in the authentication application and authentication server log files. A search of authentication requests containing the test account ID was conducted on the database of the performance monitoring tool. The data retrieved from the database was split into two data sets, one containing data from the authentication application log file and the other containing data from the authentication server log file. Analysis of the first data set revealed a large number of failed authentication requests from date-time stamp t_0 until date-time stamp t_1. Note that the authentication application does not lock user accounts after multiple failed login attempts.

Next, a search was conducted of the web server log file data between data-time stamps t_0 and t_1. Since the brute force attack was known to have originated from one location (under the conditions of the experiment), a search was performed for a requesting address that issued a large number of authentication resource requests (of the same order of magnitude as the number of authentication requests in the first data set). The brute force attack was expected to produce a large number of

authentication resource requests compared with the normal number of requests – only one requesting address was discovered to have satisfied this condition. The attacker confirmed that this address matched that of the workstation used to conduct the attack.

The experiment was conducted four times, with each attack launched from a different location. In every instance, the origin of the attack was determined correctly.

4.3 Categorizing User Activity

A second experiment analyzed the log file data to construct a profile of user login activity. The user login activity profile could then be used to identify abnormal login activity. Abnormal activity could indicate a possible security breach, e.g., a malicious entity uses stolen authentication credentials to impersonate a legitimate user.

A subset of users was originally chosen to create the baseline. However, we discovered that this was not a good representation of user activity because user login activity appeared to be different for different users. Therefore, a decision was made to build a profile for each user in a smaller subset of the test group. The profiles were then used to check for anomalous user login activity.

The tests showed that the user login profiles did not take into account divergent login activity over time. In particular, login activity was not uniform over the year (e.g., academic users tend to login less frequently at the beginning of the term than towards the end of the term). We concluded that, while it is possible to use log file data to generate user login activity profiles, more information is necessary to account for sporadic changes in login activity (e.g., enrolled courses, assignment schedules and examination schedules of the profiled users).

5. Conclusions

Combining system monitoring and digital forensic readiness in a single system is a promising concept that facilitates systems administration, performance evaluation and maintenance as well as forensic investigations. A proof-of-concept system employing a performance monitoring tool developed for an enterprise system to collect data from system and application log files in a forensically sound manner demonstrates the utility and feasibility of the combined approach.

Our future work will focus on implementing the approach in other applications with the goal of collecting richer forensic data. The extended data sets can also be used to support other types of analysis such as user activity profiling and intrusion detection.

Acknowledgements

This research was supported by the National Research Foundation of South Africa under a South Africa/Germany Research Cooperation Program.

References

[1] D. Harrington and J. Schoenwaelder, Transport Subsystem for the Simple Network Management Protocol (SNMP), RFC 5590, Internet Engineering Task Force, Fremont, California, 2009.

[2] M. Kohn, J. Eloff and M. Olivier, UML Modeling of Digital Forensic Process Models (DFPMs), Technical Report, Information and Computer Security Architectures Research Group, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2009.

[3] Oxford University Press, Oxford Dictionaries, Oxford, United Kingdom (`oxforddictionaries.com`), 2012.

[4] G. Palmer, A Road Map for Digital Forensic Research, DFRWS Technical Report DTR-T001-01 Final, Digital Forensic Research Workshop, Utica, New York (`www.dfrws.org/2001/dfrws-rm-final.pdf`), 2001.

[5] R. Rowlingson, A ten step process for forensic readiness, *International Journal of Digital Evidence*, vol. 2(3), 2004.

[6] T. Ylonen, The Secure Shell (SSH) Protocol Architecture, RFC 4251, Internet Engineering Task Force, Fremont, California, 2006.

[7] K. Zeilenga, Lightweight Directory Access Protocol (LDAP) Directory Information Models, RFC 4512, Internet Engineering Task Force, Fremont, California, 2006.