Chapter 13

FORENSIC PROFILING SYSTEM

P. Kahai, M. Srinivasan, K. Namuduri and R. Pendse

Abstract Hacking and network intrusion incidents are on the increase. However, a major drawback to identifying and apprehending malicious individuals is the lack of efficient attribution mechanisms. This paper proposes a forensic profiling system that accommodates real-time evidence collection as a network feature to address the difficulties involved in collecting evidence against attackers.

Keywords: Forensic profile, intrusion detection, alert, probe, audit trail

1. Introduction

Most organizations secure their networks using encryption technologies, network monitoring tools, firewalls and intrusion detection and response mechanisms. Despite all these security measures, compromises occur daily. Evidence collection, IP traceback and identification of attackers are as important as effective intrusion detection when a network attack takes place. However, while intrusion detection systems (IDSs) are fairly mature, very few tools exist for IP traceback and attacker identification. Prosecution is hampered by the non-availability of evidence in cases involving expert hackers and jurisdictional constraints on law enforcement. This paper proposes a forensic profiling system that accommodates real-time evidence collection as a network feature to address the difficulties involved in collecting evidence against attackers.

2. Related Work

Collaboration between intrusion detection and response systems has been the focus of recent research. MIRADOR [3] implements cooperation between multiple intrusion detection systems through a cooperation module. The cooperation module, CRIM [4], provides the interface for

alert clustering, alert merging and alert correlation. The Common Intrusion Specification Language (CISL) [7] presents a language for communication between intrusion detection systems in a network.

Alert aggregation and alert correlation have been investigated by several researchers [3–5, 10, 13]. The clustering of similar intrusion alerts is discussed in [3, 13], but the authors do not emphasize the underlying relationships between alerts. Also, most alert correlation methods are restricted to known attack scenarios. A formal framework for alert correlation and detection of multi-stage attacks is described in [10]. Alert correlation is performed if the consequences of a previous alert serve as prerequisites for the current alert. But the alerts do not confirm the possible consequences. For example, the detection of a buffer overflow attack does not imply that the attacker was successful in acquiring root privileges. In order to determine if the attack was indeed successful, participation from other network components is essential. This paper proposes a mechanism for real-time forensic evidence collection, where each network element that detects suspicious activity provides evidence in the form of log entries indicative of the activity.

3. Forensic Profiling System

The Forensic Profiling System (FPS) engages a client-server architecture. Each node in the network, referred to as a *forensic client*, is capable of detecting an anomaly, upon which it warns a central *forensic server* about the anomaly in the form of an alert. All the forensic clients participate in distributed intrusion detection and, therefore, maintain logs. A forensic client can be a router, signature analyzer, IDS, firewall or network host. The FPS logical architecture, presented in Figure 1, shows the interactions between the forensic server and forensic clients using alerts and probes.

Upon detecting an incident, a forensic client sends an alert to the forensic server along with evidence (logs) indicative of the incident. The forensic server correlates the alerts and the responses to any probes it issues and builds a forensic profile. The generation of probes is dependent on the forensic profile database maintained by the forensic server. The database contains information about known/investigated attacks in the form of descriptors called *forensic profiles*.

3.1 Forensic Profiles

A forensic profile is a structure that provides information about an attack in a succinct form. In its nascent state, a forensic profile, is based on knowledge about an attack; it is a collection of alerts that provides

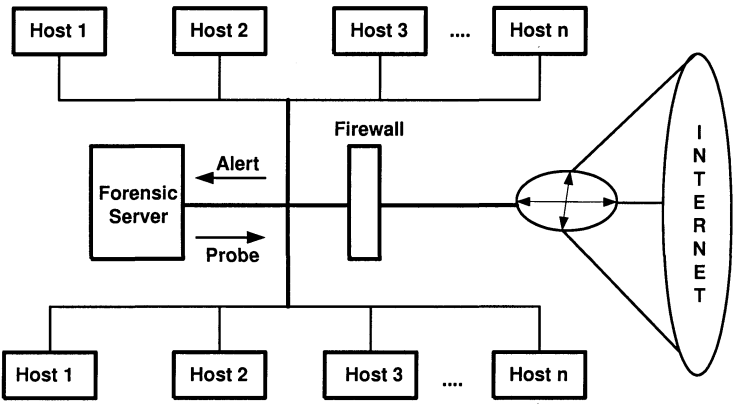

Figure 1. Forensic Profiling System (FPS) architecture.

an indication of the attack. An attack is composed of a series of inter-related events. A subset of these events might be common to several attacks. Thus, a stand-alone event does not give complete information about an attack. In order to ascertain that a particular attack has occurred, a certain minimum number of events must be detected. A profile is a structure/descriptor that defines an attack in terms of its related events; it provides relevant information about the attack in terms of its associated alerts.

The *passive state* of a network is defined as the state wherein the network is involved in normal activities that do not impact its security. In the passive state, the forensic server maintains a database of *passive profiles* of all known attacks. The passive profile is partial because it provides static and general information about an attack. The detection of an event generates an alert. A passive profile may become active when an alert is generated. The passive profiles, which contain a match for the alert generated, are considered to be active. Figure 2 shows the relationship between the Alert X received from a forensic client with the forensic profile database, which is required to shortlist all active profiles. Alert X is a subset of the alerts associated with Forensic Profiles 1 and 3. Therefore, the Profile Descriptors 1 and 3 are activated.

As an alert is a parameter of a profile, the forensic server searches for a match for the alert in the passive profiles and generates a stack of *active profiles*. To select a specific active profile, the forensic server queries the other network entities for a similar kind of event by transmitting a probe to all the forensic clients. If a forensic client responds with information pertinent to one or more active profiles, the forensic server reduces the

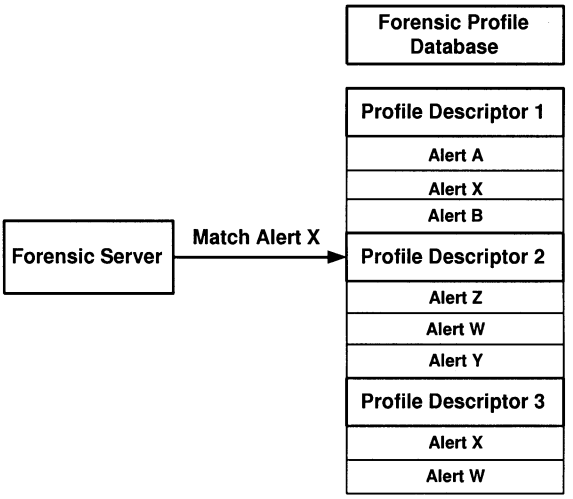

Figure 2. Active profile descriptors created by an alert.

active stack accordingly. This process recurses until the entire attack is detected. The forensic profile is thus constructed from the detection of the first alert to the detection of the attack.

3.2 Client-Server

The forensic server coordinates the activities of forensic clients. A forensic client continuously searches for anomalous activity and listens to probes from the server through agent_alert and agent_probe, respectively. The detection of a security incident involves *active monitoring* and *active parsing*. Active monitoring involves observing performance parameters such as CPU utilization and event-log intensity of the client and checking for discrepancies. Active parsing involves continuously scanning entries in the log files and history files and checking for suspicious entries (keywords), such as authentication failure, access denied and connection failure. An alert has *When-Subject-Object-Action* fields. *Subject* is the forensic client that triggers the alert, *Action* specifies the event, and *Object* is the network element on which the *Action* occurs. Events capable of triggering alerts are listed in Table 1.

The forensic server generates two kinds of probes, Check_Probe and GetLog_Probe. Check_Probe checks for suspicious activity in relation to an earlier alert received by the server. If the forensic client responds with a NULL packet to the Check_Probe then the server does not send a GetLog_Probe. Otherwise, the forensic server sends GetLog_Probe to

Table 1. Suspicious events capable of triggering alerts by forensic clients

Event	Alert
Change in CPU Utilization	*CPU_Util{SubjectIP, Up/Lo Flag, Percent}*
Frequency of log entries	*Log_Intensity{SubjectIP, FreqLogEntries}*
Increased Memory Utilization	*Mem_Util{SubjectIP, Percent}*
N/w utilization	*BandWidth{ SubjectIP, CurrentUtil}*
Login Denied	*DeniedLogin{SubjectIP, Username, Remote/local, AttemptNo}*
Invalid IP (reserved and not used)	*InavalidIP{SubjectIP, Invalid IP}*
System File deletion	*DeleteFile{SubjectIP, FileName}*
Change of privileges for System Log	*Chmod{SubjectIP, Syslog}*
Change of privileges for History File	*Chmod{SubjectIP, Hist}*
Connection Failed	*FailCon{SrcIP:SrcPort,DestIP:DestPort, AttemptNo}*
Upload File	*FileUpload{SubjectIP (Server), FileName}*
Unsetting History File	*UnsetHist{SubjectIP}*
Change attributes of history file	*Chattr{SubjectIP, Hist}*
Kill logging daemon	*Kill{SubjectIP, LogD}*
Kill kernel log daemon	*Kill{SubjectIP, KlogD}*

receive the log entries for that particular event. Figure 3 shows the Probes A, B and W corresponding to the alert and the forensic profiles shown in Figure 2.

The mechanism discussed in Section 3.1 is applicable to unknown attacks. An unknown attack does not have a passive profile. However, since attacks have common events that trigger them, the alerts generated would be used to save the log entries in the forensic server that can be later used to trace the attacker and corroborate evidence. Intrusion detection systems, which are based on attack signatures, are unable to track illicit activity caused by a new or unknown attack. The forensic profiling system deals with this problem by creating an unstructured profile. If an unsolicited event does not match any of the known profiles, an unstructured profile is constructed with all the alerts generated in the same region and time span. This ensures that even if the attack was not stopped, evidence related to the activity is collected and saved.

3.3 Active Event Monitoring

Active monitoring is a process that continuously checks for variations in the intensity of events. Exponentially weighted moving average (EWMA), a statistical process control technique, is used to detect drift

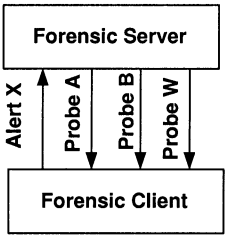

Figure 3. Sample forensic server/client communications.

in a parameter being monitored [15]. The decision regarding the state of control of a process at any time instant depends on the EWMA statistic, which is an exponentially weighted average of all the prior data and depth of memory. The EWMA control technique takes the statistic (CPU utilization, traffic intensity, log event intensity, memory utilization) that is to be monitored as argument in real time and recursively checks if the current value lies within the control limits. The control limits are determined by training data composed of usual or normal events. The testing data are interspersed with intrusive events. The events in an information system are closely related to each other. Therefore, the EWMA technique that makes use of auto-correlated data has been applied. The EWMA statistic for i-*th* observation *z(i)*, is given as:

$$z(i) = \lambda \cdot x(i) + (1 - \lambda) \cdot z(i - 1) \qquad i = 1..n \tag{1}$$

where *z(0)* is the mean of the training data, *x(i)* is i-*th* observation, n is the number of observations to be monitored and $0 < \lambda \leq 1$ is the depth of memory. The depth of memory determines the rate at which the past observations enter into the calculation of EWMA statistic. Conventionally, $\lambda = 1$ implies the most recent observation influences EWMA. Thus, larger values of λ give more weight to recent data. The depth of memory depends on the parameter being monitored, i.e., greater depth into the history of events is required for event log intensity than for CPU utilization. For our calculations, λ lies between 0.2 and 0.3 for log intensity measurement and 0.8 for CPU utilization. An anomaly is detected if *z(i)* falls outside the control limits and an alert is generated by the forensic client to the forensic server.

3.4 Active Parsing Enabling Mechanisms

Most alerts generated by the forensic clients are based on parsing log files. Thus, the efficacy of FPS depends on maximizing the logging mechanisms of forensic clients.

Continually parsing a history file helps identify the execution of suspicious commands like chattr, chmod for critical files such as log files and the history file itself. User activity logging can be configured on a Linux machine by making use of the process accounting package. All the commands used at the console are logged into a binary file. This provides more information about command execution than the history file in terms of the user who executed a command, CPU time, connection times for each user, etc. Network monitoring tools must be deployed for logging network activities. Firewalls must be configured to log connection failures, and servers must be configured to log actions specific to the services provided by the server.

4. Case Study

This section presents a case study involving a WU-FTP attack [2] on a network, interjected by the alerts and probes generated by FPS if it were part of the network. To exploit the Globbing vulnerability associated with Version 2.6.1 or earlier of a WU-FTP server, the attacker should have authenticated or anonymous access to the FTP server. Next, the attacker should be able to create a buffer overflow and then execute a process that installs a backdoor/Trojan/rootkit. The following subsections describe the victim network and the trail of events associated with the attack.

4.1 Victim Network

The internal network of the company and its DMZ setup is well-designed from a security perspective. The DMZ consists of the standard set of network servers (web, email, DNS servers and a dedicated FTP server for distributing hardware drivers for the company inventory). Two firewalls are used, one separating the DMZ from the Internet and the other separating the DMZ from the internal network (LAN). Two network IDSs are part of the DMZ setup. Individual firewalls are installed in each of the DMZ machines. No connections are allowed from the DMZ to either the Internet or to the LAN. Also, no connections are allowed between the DMZ machines themselves. An outside machine may only connect to a single port of each DMZ host. Each DMZ machine runs a host-based firewall.

The forensic server is an integral part of FPS as it maintains the forensic profile database. Since the focus is on the WU-FTP attack, its forensic profile descriptor is shown in Figure 4.

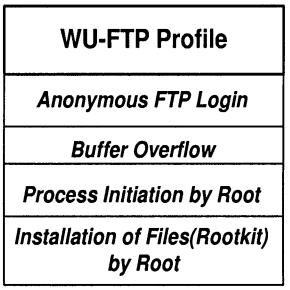

Figure 4. Forensic profile for the WU-FTP attack.

All the network components are configured as forensic clients by installing agents. Unauthorized network connections are detected by firewalls, which issue alerts in the following format:

$$FailCon\{SourceIP{:}Port, DestinationIP{:}Port, AttemptNo\}$$

4.2 Attack Progression

The forensic team was notified after a customer was unable to connect to the company's FTP server. It was subsequently discovered that the server's operating system had been deleted. Live forensic analysis of the machine could not be performed as the server had crashed while it was unattended. The log files of the FTP server also could not be recovered as the syslog was not configured for remote logging. Therefore, it was only possible to analyze the hard drive of the FTP server and logs retrieved from the IDS and firewalls. Analyzing the 20GB hard drive for forensic evidence was deemed to be too time consuming.

The forensic investigation revealed that the IDS detected the WU-FTP attack on Apr 1 02:29:45. The FPS response mechanism prompted the IDS to send the following alert to the forensic server.

- Alert generated by IDS to the forensic server:

When	Subject	Object	Action
Apr 1 02:29:45	IP Addr IDS	IP addr FTP server	WU FTPD attack

The forensic server reacted to the alert by issuing probes. Alerts can be generated by two or more forensic clients simultaneously depending on the type of event detected by the clients. Suspicious activity detected by a forensic client will not, on its own, determine the progression of an attack. But an IDS can detect an attack as a whole. Thus, the implementation of the forensic profiling system differs in the steps followed in tracking the attack depending on the client that triggers the alert. (The

alerts to the server may be sent in a random fashion.) We examine the flow of events when the attack was detected by the IDS.

Check probes were generated simultaneously because, although they are related from the point of view of the attack and occur in a chronological order, they are independent of each other. The probes generated by the forensic server corresponded to the alerts contained in the descriptor for the WU-FTP attack. Therefore, the Check_Probes sent were:
(i) Check_Probe sent to the FTP server:

Dest Addr	CheckFlag	Time
IP Addr FTPserver	FTPLogin	Apr 1 02:29:45

If the FTP server had sent a NULL packet, this would have indicated that no one was logged into the FTP server at the time. Otherwise, the FTP server would have responded by providing the IP addresses that were logged in at the time. The server went over the logs it captured through the forensic clients and scanned for a match for the IP addresses sent by the FTP server. A matched IP address is suspicious because the forensic server has logs only for suspicious activities. The forensic server then sent GetLog_Probe to the FTP server which specified the suspicious IP address as the keyword. The following alert showed that 192.1.2.3 is a suspicious IP address.

Dest Addr	GetLogFlag	Keyword	Time
IP addr FTP Server	Set	192.1.2.3	Apr 1 02:29:45

The log fragments that corroborated the alert recovered from the FTP server are presented below.

FTP System Logs:

Apr 1 00:08:25 ftp ftpd[27651]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 00:17:19 ftp ftpd[27649]: lost connection to 192.1.2.3 [192.1.2.3]
Apr 1 00:17:19 ftp ftpd[27649]: FTP session closed
Apr 1 02:21:57 ftp ftpd[27703]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 02:26:13 ftp ftpd[27722]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], mozilla@
Apr 1 02:29:45 ftp ftpd[27731]: ANONYMOUS FTP LOGIN FROM
192.1.2.3 [192.1.2.3], x@

(ii) Check_Probe sent to the IDS:

Dest Addr	CheckFlag	Time
IP Addr IDS	Buffer Overflow	Apr 1 02:29:45

If the response sent by the IDS to the server contained a NULL packet, a buffer overflow condition would have been negated. Otherwise the forensic server would have sent the GetLog_Probe.

(iii) Check_Probe sent to the FTP server:

Dest Addr	CheckFlag	Time
IP Addr IDS	Process Execution + Root Access	Apr 1 02:29:45

If the response sent by the FTP server to the forensic server contained a NULL packet, this would have implied that no script was running on the server. Otherwise, the forensic server would have sent the GetLog_Probe. The forensic server would have continuously sent GetLog_Probes to the FTP server. If the FTP server had crashed as a result of the attack, it would have stopped serving the GetLog request initiated by the forensic server.

After gaining access to the FTP server, the attacker tried to connect to his machine (192.1.2.3), which was not allowed. Also, the attacker attempted to connect to the mail server. This is implied by the following FTP connection logs.

FTP Connection Logs:

Apr 1 02:30:04 ftp ftpd[27731]: Can't connect to a mailserver.
Apr 1 02:30:07 ftp ftpd[27731]: FTP session closed

The corresponding alert indicated that an unauthorized network connection attempt was generated by the FTP server.

- Alert generated by FTP server to the forensic server for connection failure to mail server:

When	Subject	Object	Action
Apr 1 02:30:04	IP Addr FTP Server	IP Addr Mail server	FailCon

A similar alert was generated by the firewall.

The attacker was able to gain root access, upload a file and later execute a script. This can be inferred from the FTP transfer logs.

FTP Transfer Logs:

Mon Apr 1 02:30:04 2002 2 192.1.2.3 262924 /ftpdata/incoming
/mount.tar.gz b _ i a x@ ftp 0 * c

The alert generated by the FTP server to the forensic server, indicative of uploading a file with root privileges, is shown below.

- Alert generated by the FTP server to the forensic server indicating file upload:

When	Subject	Object	Action
Mon Apr 1 02:30:04 2002	IP Addr FTP Server	mount.tar.gz	FileUpload

The attacker was able to upload files on the FTP server as the FTP server was world writable. This file is the suspected rootkit. The attacker later deleted the operating system causing the FTP server to crash.

5. Conclusions

This paper proposes a forensic profiling system (FPS) for real-time forensic evidence collection. A dedicated forensic server maintains an audit trail embedded in a forensic profile. Because FPS keeps track of anomalous activities in a network, the time spent on filtering system log files during a forensic investigation is drastically reduced. FPS also makes it easier to retrieve the logs of crashed hosts as the hosts can send log entries associated with alerts to the forensic server. Since all attacks have a general commonality, unknown attacks can be tracked by the forensic sever on the basis of the alerts generated by forensic clients. A detailed investigation of attacks is required to construct forensic profiles. Also, it is necessary to evaluate the overhead involved in active parsing and monitoring.

References

[1] J. Barrus and N. Rowe, A distributed autonomous agent network intrusion detection and response system, *Proceedings of the Command and Control Research Technology Symposium*, pp. 577-586, 1998.

[2] A. Chuvakin, FTP Attack Case Study, Part I: The Analysis (www.linuxsecurity.com/feature_stories/ftp-analysis-part1.html) 2002.

[3] F. Cuppens, Managing alerts in a multi intrusion detection environment, *Proceedings of the Seventeenth Annual Computer Security Applications Conference*, 2001.

[4] F. Cuppens and A. Miège, Alert correlation in a cooperative intrusion detection framework, *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[5] H. Debar and A. Wespi, Aggregation and correlation of intrusion detection alerts, *Proceedings of the Fourth International Workshop on Recent Advances in Intrusion Detection*, pp. 85-103, 2001.

[6] M. Huang, R. Jasper and T. Wicks, A large-scale distributed intrusion detection framework based on attack strategy analysis, *Proceedings of First International Workshop on Recent Advances in Intrusion Detection*, 1998.

[7] C. Kahn, D. Bolinger and D. Schnackenberg, Common Intrusion Detection Framework (www.isi.edu/gost/cidf/), 1998.

[8] P. Ning, Y. Cui and D. Reeves, Constructing attack scenarios through correlation of intrusion alerts, *Proceedings of the Ninth ACM Conference on Computer Security*, 2002.

[9] P. Ning, X. Wang and S. Jajodia, A query facility for the common intrusion detection framework, *Proceedings of the Twenty-Third National Information Systems Security Conference*, pp. 317-328, 2000.

[10] P. Ning, X. Wang and S. Jajodia, Abstraction-based intrusion detection in distributed environments, *ACM Transactions on Information and System Security*, vol. 4(4), pp. 407-452, 2001.

[11] P. Porras and P. Neumann, EMERALD: Event monitoring enabling responses to anomalous live disturbances, *Proceedings of the Twentieth National Information Systems Security Conference*, pp. 353-365, 1997.

[12] K. Shanmugasundaram, N. Memon, A. Savant and H. Bronnimann, ForNet: A distributed forensics network, *Proceedings of the Second International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security*, 2003.

[13] A. Valdes and K. Skinner, Probabilistic alert correlation, *Proceedings of the Fourth International Workshop on the Recent Advances in Intrusion Detection*, 2001.

[14] J. Yang, P. Ning and X. Wang, CARDS: A distributed system for detecting coordinated attacks, *Proceedings of the IFIP TC11 Sixteenth Annual Working Conference on Information Security*, 2000.

[15] N. Ye, S. Vilbert and Q. Chen, Computer intrusion detection through EWMA for autocorrelated and uncorrelated data, *IEEE Transactions on Reliability*, vol. 52(1), pp. 75-81, 2003.