Paper

Detecting Security Violations Based on Multilayered Event Log Processing

Przemysław Malec, Anna Piwowar, Adam Kozakiewicz, and Krzysztof Lasota

Research and Academic Computer Network (NASK), Warsaw, Poland

Abstract—The article proposes a log analysis approach to detection of security violations, based on a four layer design. First layer, named the event source layer, describes sources of information that can be used for misuse investigation. Transport layer represents the method of collecting event data, preserving it in the form of logs and passing it to another layer, called the analysis layer. This third layer is responsible for analyzing the logs' content, picking relevant information and generating security alerts. Last layer, called normalization layer, is custom software which normalizes and correlates produced alerts to raise notice on more complex attacks. Logs from remote hosts are collected by using rsyslog software and OSSEC HIDS with custom decoders and rules is used on a central log server for log analysis. A novel method of handling OSSEC HIDS alerts by their normalization and correlation is proposed. The output can be optionally suppressed to protect the system against alarm flood and reduce the count of messages transmitted in the network.

Keywords*—*HIDS, log analysis, NIDS, syslog.

1. Introduction

Events occurring in the operating system, like software installation, managing system services, as well as successful and failed login attempts, are preserved real-time in the form of logs. Every log stores data regarding its origin, priority and time of appearance, which allows use of event logs as a reliable source of information when building systems for alerting about security violations. Raising alarm after detecting every single malfunction would lead to frequent false positives. That is why receiving and correlating data from multiple event log sources would increase accuracy of detection and allow to reveal violations with more complex indicators.

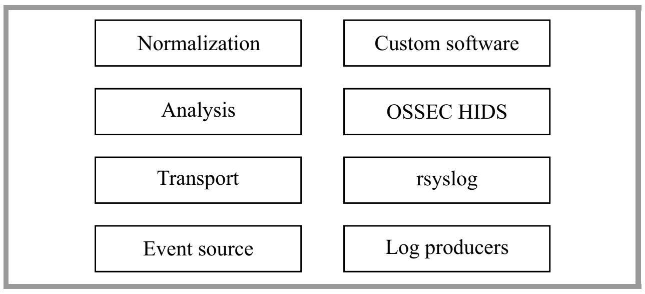

Fig. 1. Multilayered event log processing.

The design of the log-based system for detecting security violations consists of four layers, presented in Fig. 1.

The bottom layer, named event source, specifies the log sources in Linux and network environment relevant in the process of detecting events. The essential information may originate from typical system services and network devices, as well as from security dedicated services like audit and integrity check tools.

Transport layer is responsible for collecting log messages from various sources and passing them to the log-collecting server, where the analysis is done. The transport must guarantee confidentiality and transmitted data integrity, achieved by using *rsyslog* software [1]. This layer secures a copy of all incoming logs (crucial in security log management according to the guidelines [2]), which enables discovery of data tampering attempts. Preserving logs in remote localization enables incident reconstruction even after unrecoverable machine failure [3]. Moreover, preserving three timestamps for every log (generation, server reception, database insert) can indicate server downtime or communication disruption, which can be relevant for further investigation.

The role of the analysis layer is to generate alerts based on incoming log entries by decoding key information and filtering events that are relevant, while detecting malicious behavior in the network. The first analysis is executed by *OSSEC HIDS* [4] engine with a set of custom-written decoders and rules.

The need to decrease the number of repetitive alerts was widely discussed in [5], [6] and some of the existing strategies of alarm suppression were presented in [7]. The normalization layer, implemented by dedicated software, examines the output of the analysis layer and suppresses excess alerts. The need to combine many sources of information expressed in [8] is satisfied by performing nonlinear correlation to detect more complex attacks. As a result, this layer creates alarms that are normalized to a protocol, and can be used by security system consumer.

2. Event Source Layer

2.1. Log Sources

Log messages containing knowledge about events taking place in the operating system or the network can be obtained from various process sources. There are several services and modules that gather information about security events and store it in the form of logs, shown in Fig. 2. This article focuses on events generated by following sources:

- *auditd*,
- Advanced Intrusion Detection Environment (AIDE),
- *sshd*,
- *OSSEC rootcheck*,
- *racoon*,
- *iptables*,
- network devices,
- event logs coming from dedicated processes related with the system's security.

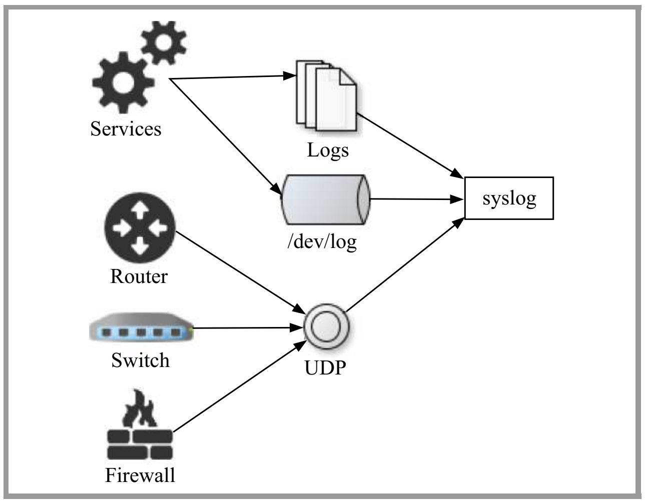

Fig. 2. Event sources.

auditd [9] is based on pre-configured rules and generates log entries recording a large variety of information about the events occurring in the system. According to [10], auditd can discover policy violations by monitoring file activities and collecting system calls.

AIDE [11] is a file and directory integrity checker. It creates a database from the state of the system, register hashes, modification times, etc. This database is later used to verify the integrity of files by comparing it against the real status of the system. All of the usual file attributes can also be checked for inconsistencies. AIDE generates syscall ANOM_RBAC_INTEGRITY_FAIL, when change in the monitored file structure is detected and received by auditd from kernel.

sshd [12] is an OpenSSH Daemon, which provides secure encrypted communications between two untrusted hosts over an insecure network. It can refer to Pluggable Authentication Modules (PAMs) [13], which provide a common authentication framework for applications and services in a Linux system. The errors in communication can be a valuable source for log analysis.

OSSEC's rootcheck is an OSSEC HIDS module for rootkit detection, which runs at regular intervals querying the system for information and comparing the results with a list of known rootkits and trojans. When the rootcheck module finds discrepancies in information about a file, a process, port or network interface, it will raise an alert about a suspected rootkit.

racoon [14] is an Internet Key Exchange (IKE) daemon for automatically keying IPsec connections to establish safe associations between hosts. Reported errors in communication can be used to detect suspicious behavior of nodes.

iptables [15] is used as a firewall to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel.

Network devices (i.e., routers, switches, firewalls) send information about their activity in UDP messages.

2.2. Log Formats

Log records are an essential source of information and can be written by a process to a dedicated text file, although the majority of logs are a product of sending data to local /dev/log socket, from where the messages are received by a log collecting service called syslog. All information collected by syslog should by written in a simple syslog format described in RFC 5424 [16] with a such structure:

```
<PRI> TIMESTAMP HOSTNAME APP-NAME: MSG
```

The value of the PRI part is assigned based on two properties, which are severity and facility. Severity is a numerical value of event priority and varies from 0 (emergency) to 7 (debug). Facility value depends on the log supplier, where different kinds of system services have taken their own codes, e.g., kernel messages (kern) have 0 facility value, authentication events (auth) have facility value equal to 4, etc. In addition, every log contains the timestamp of the logged event, hostname of the machine which produced the message, program source of the event (optionally with PID in brackets) and message content of registered log.

An example of a syslog protocol log is shown below:

```
Jul 7 14:37:10 pl.bipse.wil.si.kbk1
sshd[15308]:  Accepted password for root from
192.168.56.1 port 56440 ssh2
```

Audit has an independent log processing system, which writes events to /var/log/audit/audit.log file and also provides a rotation mechanism for its journals. However, with the usage of audisp daemon the audit output can be redirected to the standard /dev/log socket for further processing. The audit log contains information about the type of registered event (system call, login information, etc.), its timestamp and a list of audit event fields suitable for given occurrence.

The example of audit log is as follows:

```
type=SYSCALL msg=audit(1441374914.091:1975):
arch=c000003e syscall=82 success=yes
exit=0 a0=1668900 a1=4a4fa8 a2=1647c00
a3=6d617473656d6974 items=5 ppid=1 pid=1840
auid=4294967295 uid=0 gid=0 euid=0 suid=0
fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none)
ses=4294967295 comm="NetworkManager"
exe="/usr/sbin/NetworkManager" key="LINKING"
```

3. Transport Layer

3.1. Open Source Systems for Log Processing

A mechanism for centralized logging can be set up by configuring existing open source solutions for log processing. The leading software utilities are *syslog-ng* Open Source Edition, developed by BalaBit IT Security Ltd.[17] and *rsyslog* from Rainer Gerhards and Adiscon.

The *syslog* implementation used in a security system should meet the requirements described in [18], such as high availability of service and confidentiality and integrity of transmitted data. Both of the chosen solutions fulfill these assumptions and offer the ability to send and collect remote log messages by encrypted transmission using the Transport Layer Security (TLS) mechanism. Another key feature that both *rsyslog* and syslog-ng provide is database support. In addition, syslog-ng represents a highly customizable solution that can gather information from many different sources beside the standard operating system events, e.g. additional text files and the content of binary files such as /var/log/btmp, which contains records of failed user login attempts). The syslog-ng's database support is flexible and allows storing logs in daily tables without additional administrational procedures. The main disadvantage of syslog-ng is its fix-sized queue mechanism, which does not guarantee avoiding log loss during server unavailability in case of client's queue overflow.

On the other hand, rsyslog is capable of collecting standard system messages and events written to external text files and transferring them via Reliable Event Logging Protocol (RELP) to the central log server, ensuring zero message loss among the network nodes. rsyslog also supports disk buffers for not forwarded log messages, which protects log journals from loss and inconsistency.

For the purposes of the designed system rsyslog was chosen to create the centralized log processing system due to the reliability of its built-in mechanisms for log transfer between the hosts.

3.2. Proposed Architecture of the Transport Layer

The transport layer is based on client-server architecture. On every client machine data is collected from typical sources, such as the /dev/log socket (including the output of audispd redirecting audit events), system services like AIDE or sshd. Moreover, clients can gather information from network devices by receiving UDP messages and passing the relevant data further. Collected information is sent to the log server via the RELP communication, which guarantees no message loss. RELP mechanism uses a Transmission Control Protocol (TCP) connection, which is an advantage over plain syslog communication, which uses User Datagram Protocol (UDP) as a default.

The server machine receives messages from remote clients and processes them next to the logs from the local sources described above. Remote logs are written to separate files in catalogues named after client's hostnames, reflecting the file structure on origin machine. The copy of all messages is sent to the database, from where logs can easily be accessed with SQL queries. The database output stores all information included in the log, server receive timestamp and database insert time as well. The rsyslog, based on properties from syslog header, filters messages and forwards them to the named pipe /dev/ossec, from where they are received by OSSEC HIDS analyzer. The flow of information in the system is displayed in Fig. 3.

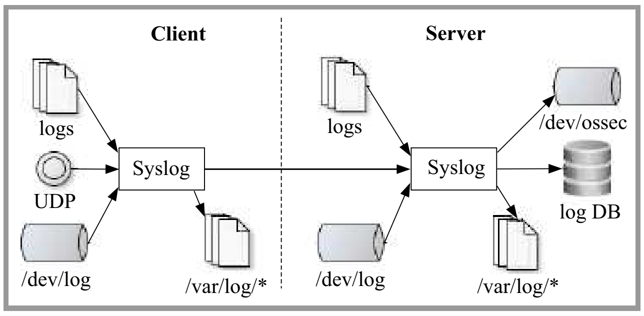

Fig. 3. Event log flow.

4. Analysis Layer

OSSEC HIDS is a platform to monitor the status of network elements. It offers the functionality of Host Intrusion Detection System (HIDS), Security Information and Event Management (SIEM), log monitoring, rootkit detection and checking the integrity of system files. It is frequently used as a part of more complex security systems, due to its flexibility and facility in adapting to own needs [19], [20]. In the proposed architecture OSSEC HIDS with custom rules and decoders is used to build a real-time log analyst monitor. This approach enables correlation of events from every node in the network. The stages of the analyst process are shown in Fig. 4.

Process steps are as follows:

- Pre-decoding – as mentioned, *rsyslog* is used as a transport channel for logs, which adds a syslog header. This step decodes the syslog format and pulls information about hostname and time from log;

- Decoding – custom decoders extract information based on program name, relevant for event type. It detects data like login name, address and ports for source and destination;

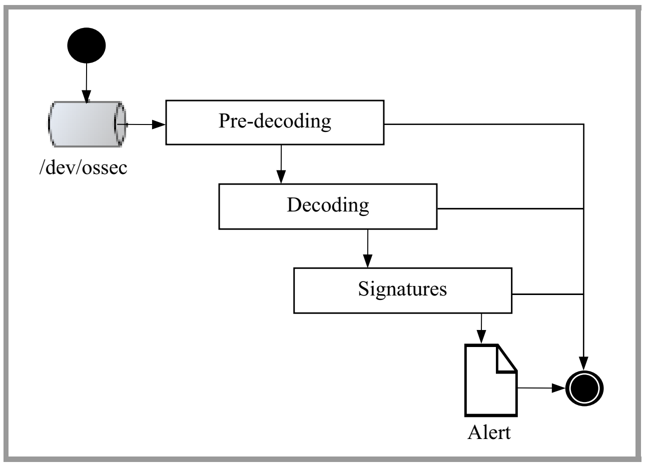

Fig. 4. OSSEC HIDS flow.

- Signatures and alerting – a set of custom rules is matched to the log event. Composite rules correlate hostnames, IPs, users etc. Every rule has an assigned priority level. When a single event matches more than one rule, the generated alarm has the priority of the rule with the highest level or with the level of the first matched rule when levels have the same value. If the incident violates adopted policies, an alarm is raised. Generated alarms precisely define the detected security events.

4.1. Pre-decoding

OSSEC HIDS recognizes different log formats. In the proposed system, the most common syslog format is used. Based on the syslog format, specified fields are decoded, such as:

- hostname – DNS/IP address of component which originated the event,
- time – time of the event from the element,
- message – message which is used in analysis,
- program – information which process generated the event.

Table 1 shows the result of the pre-decoding of a sample log.

Table 1
The result of the pre-decoding of a sample log

Time	Jul 17 08:34:40
Hostname	pl.bipse.nask.element2
Program	sshd[19721]:
Message	Accepted keyboard-interactive/pam for root from 192.168.56.1 port 51499 ssh2

4.2. Decoding

Based on the results of pre-decoding, the field named program determines the process from which the log comes. This approach minimizes the number of decoding attempts from the various decoders to only those that meet the criteria. The matching decoder will later be used to decode relevant information, for example the source IP address of the user. The decoding process consists of three stages:

- decoder selection,
- log content matching using regular expressions,
- decoding declared fields.

Example of decoding an *sshd* log is shown in Fig. 5. Based on process name sshd, the sshd decoder is used in the first stage. This decoder has a set of sub-decoders, that are used in regular expressions matching to determine which of them can decode specific field. It is possible to use several decoders simultaneously. Figure shows that the sshd-success decoder has been selected. The last stage presents that fields root and IP address were chosen and decoded as USER and SRCIP (names of fields used in the inner OSSEC HIDS logic).

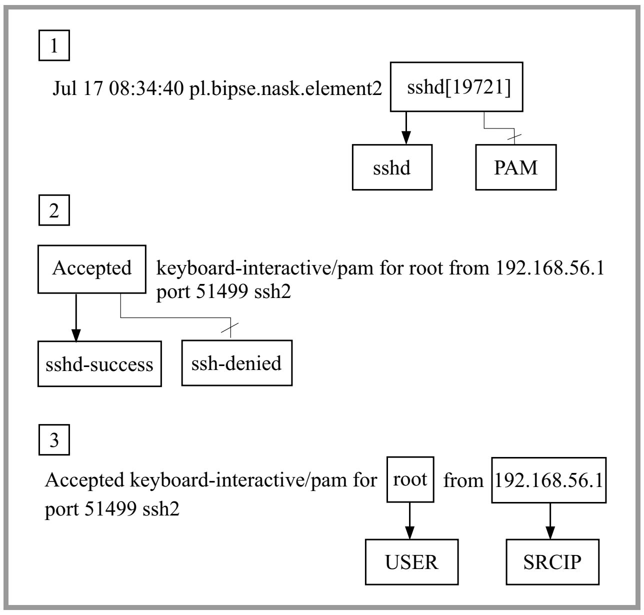
1
Jul 17 08:34:40 pl.bipse.nask.element2
sshd[19721]
sshd
PAM
2
Accepted
keyboard-interactive/pam for root from 192.168.56.1 port 51499 ssh2
sshd-success
ssh-denied
3
Accepted keyboard-interactive/pam for port 51499 ssh2
root
from
192.168.56.1
USER
SRCIP

Fig. 5. OSSEC HIDS decoding process.

4.3. Event Correlation and Signatures

OSSEC HIDS distinguishes two categories of rules that generate alarms:

- atomic – based on a single event which occurred in the system,
- composite – correlated over time, based on patterns of other logs.

Figure 6 shows the logic of matching a rule to a decoded event. Log entries are analyzed in a sub-rule only after matching its parent rule. This approach accelerates the process of analysis and minimizes the tested rules amount.

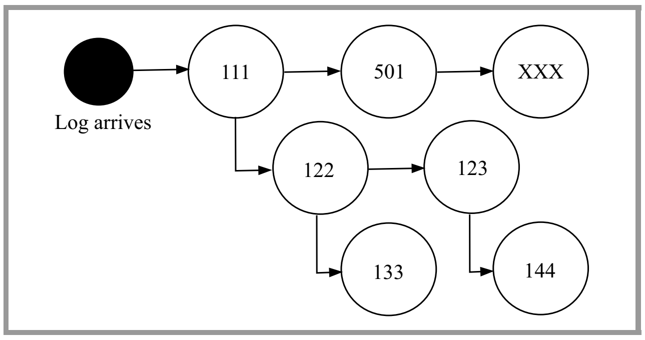

Fig. 6. OSSEC HIDS rules match.

Pessimistic number of rules to process is the sum of all parent type rules.
In the proposed solution the grouping of rules is a base for event correlation. For example, events coming from two different sources are decoded by two separate decoders and getting in two independent branches of rules. However, those rules have the same group id. While appearing individually none of the detected events generate alarms, but their correlation leads to signs of an incident being detected and raises an alarm. The grouping logic based on decoders sshd and login is shown in Fig. 7.

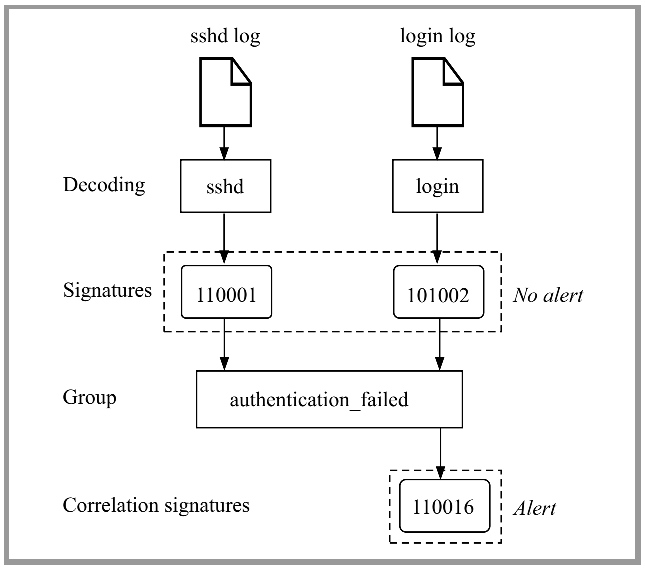

Fig. 7. OSSEC HIDS correlation and grouping logic.

4.4. Proposed Abuse Detection

Custom rules and decoders allow detection of events, which can be categorized into groups of security violation alarms like:

- file modification – files that are watched by auditd or AIDE, mostly configuration files of system or security processes,
- authentication abuse – local or remote, based on login and sshd proccess,
- rejected connection attempts – system firewall reports about rejected connection attempts, this information can be grouped and detected as a host scan,
- successful system user login – for example user apache or mysql,
- malfunctions services – for example, possible error of racoon negotiation between elements or invalidity of certificates,
- file system and hardware errors – can cause unnoticed relevant events in system, like overflow of hard drives,
- sudo abuse – system user and group modification,
- occurrence of unknown errors – events which should draw attention of the operators, those can be symptoms of attacks or reconnaissance,
- network equipment errors and events – errors and changes in the network topology are crucial information about inappropriate network activity for further investigation.

5. Normalization Layer

Previous layers process relevant event logs that can possibly generate alarms. This layer adds extra functionality, which makes events more useful and foolproof. This is performed by custom-written software. Main objectives of this layer are:

- alarm suppression,
- normalization for other security systems,
- guaranteed delivery,
- nonlinear correlation,
- rejection of irrelevant alarms.

5.1. Alarm Suppression

In the proposed solution OSSEC HIDS can aggregate and correlate events. Such aggregation can easily generate too many alerts if events incoming in a short time are a huge number of identical logs. As OSSEC HIDS suppresses identical logs, this layer suppresses OSSEC HIDS identical alarms. Each subsequent alarm which is propagated, contains the number of events summed in the suppression, so that the number of individual events is not lost. Suppression affects only the repeating alarms. The suppression time is configurable, depending on the type of event to which it refers to. This process is shown in Fig. 8. The proposed approach minimizes the amount of identical alarms sent and processed by central alarm analyzer, which results in improvement of performance without losing any information.

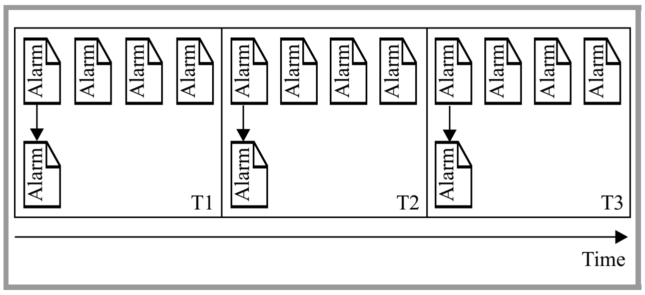

Fig. 8. OSSEC HIDS alarm suppression.

5.2. Normalization for Security System Protocol

Security systems often use their own protocols to communicate between components. Additionally, redundant information generates unnecessary network traffic, which can lead to internal Denial of Service (DoS) of security systems. Triggered alarms must be standardized to meet the needs accepted as a norm for reporting security incidents. Adopted policy may require additional data. At this point, the normalization layer enriches the message with additional data based on the configuration or correlating this event with information from knowledge bases. This is shown in Fig. 9.

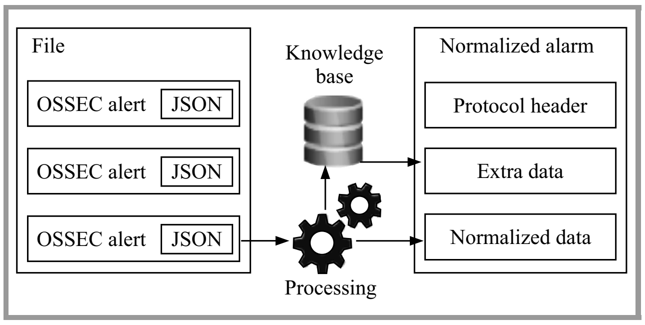

Fig. 9. Normalization for other security systems.

5.3. Guaranteed Delivery

Ability to communicate with another component that is a consumer of alerts requires normalization of forwarded information. The second step is to ensure that the detected event has been received by the central processing system. Each of the outgoing events has an identifier of an incident and its time validity. The mechanism checks whether the consumer has sent a reception acknowledgment or the validity of the message has expired. Based on this feedback information, it is possible to send a message again and guarantee delivery.

5.4. Nonlinear Correlation

The normalization layer adds a second correlation that can combine facts from various areas of events. This layer, by focusing on brokering, is able to expand the analysis based on the rules of event occurrence. This can provide historical events analysis and detection of patterns in alarms based on concept of states. Combining the facts of undesirable incidents occurring in various parts of the network and applying security modeling, could result in producing security rules that may protect other nodes, which are not endangered yet.

5.5. Rejection of Irrelevant Alarms

The normalization layer adds the ability for the complex security system to work in various modes, which may require additional alarm suppression. To reject alarms that are irrelevant, when system's state is taken into consideration, following modes can be distinguished:

- learning mode – detected incidents are a base for rule creation, which will preserve similar events from propagating in normal mode,
- reconfiguration mode – alarms are temporarily suppressed when the system is under configuration and modifications could result in false-positives,
- normal mode – every event classified as an alarm is propagated further, unless there was a corresponding rule created in the learning mode.

6. Summary

The article presented a multilayered approach to managing and handling security incidents based on event logs. Each layer presented key aspects of its functioning with examples of implementation. Use of ready-made solutions allows to focus attention on upper layers associated with event processing logic. The division of responsibilities into layers allows easier modification and implementation as part of security systems.

Acknowledgements

This work was supported by the National Centre for Research and Development (NCBiR) as a part of the research project "The system of secure IP communication provision for the power system management" (no. ROB 0074 03 001).

References

[1] Rsyslog – The rocket fast system for log proseccing [Online]. Available: http://www.rsyslog.com

[2] K. Kent and M. Souppaya, "Guide to Computer Security Log Management", National Institute of Standards and Technology (NIST) Special Publication 800-92, 2006.

[3] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge", in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Edmonton, Alberta, Canada, 2002, pp. 366–375.

[4] OSSEC documentation [Online]. Available: http://ossec-docs.readthedocs.org

[5] H. W. Njogu and L. J. Wei, "Using alert cluster to reduce IDS alerts", in *Proc. 3rd IEEE Int. Conf. Comp. Sci. Inform. Technol. ICCSIT 2011*, Chengdu, China, 2011, pp. 467–471.

[6] H. T. Elshoush and I. M. Osman, "Alert correlation in collaborative intelligent intrusion detection systems – A survey", *Appl. Soft Comput.*, vol. 11, pp. 4349–4365, 2011.

[7] T. H. Nguyen, J. Luo, and H. W. Njogu, "Improving the management of IDS alerts", *Int. J. Secur. Its Appl.*, vol. 8, no. 3, pp. 393–406, 2014.

[8] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis", *Commun. of the ACM (CACM)*, vol. 55, no. 2, pp. 55–61, 2012.

[9] auditd – security guide [Online]. Available: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security_Guide/chap-system_auditing.html

[10] "Guide to the Secure Configuration of Red Hat Enterprise Linux 5", National Security Agency, Revision 4.2, pp. 87–94, 2011.

[11] AIDE – Advanced Intrusion Detection Environment [Online]. Available: http://aide.sourceforge.net/

[12] sshd deployment guide [Online]. Available: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-OpenSSH.html

[13] Using Pluggable Authentication Modules (PAM) [Online]. Available: https://access.redhat.com/documentation/en-US/Red_Hat _Enterprise_Linux/6/html/Managing_Smart_Cards/Pluggable _Authentication_Modules.html

[14] racoon – Linux daemon [Online]. Available: http://linux.die.net/man/8/racoon

[15] iptables [Online]. Available: https://wiki.centos.org/HowTos/Network/IPTables

[16] R. Gerhards, The Syslog Protocol, RFC 5424 [Online]. Available: https://tools.ietf.org/html/rfc5424

[17] Syslog-ng – open source log management [Online]. Available: https://syslog-ng.org/

[18] K. E. Nawyn, "A Security Analysis of System Event Logging with Syslog", SANS Institute, no. As part of the Information Security Reading Room, 2003.

[19] L. Ying, Z. Yan, and O. Yang-jia, "The design and implementation of host-based intrusion detection system", in *Proc. 3rd Int. Symp. Intell. Inform. Technol. Secur. Informat.*, Jinggangshan, China, 2010, pp. 595–598.

[20] J. Timofte, "Intrusion Detection using Open Source Tools", *Revista Informatica Economică*, no. 2, vol. 46, pp. 75-79, 2008.

Przemysław Malec got his M.Sc. in Computer Science in 2015 at Polish-Japanese Academy of Information Technology. At present he is a research assistant at NASK. His main scientific interests concern network programming and information security.

E-mail: przemyslaw.malec@nask.pl
Research and Academic Computer Network (NASK)
Wawozowa st 18
02-796 Warsaw, Poland

Anna Piwowar got her M.Sc. in 2013 in Electrical and Computer Engineering from Warsaw University of Technology, Poland. At present she is a research assistant at NASK. Her main scientific interests include information security, especially intrusion detection and prevention systems.

E-mail: anna.piwowar@nask.pl
Research and Academic Computer Network (NASK)
Wawozowa st. 18
02-796 Warsaw, Poland

Adam Kozakiewicz, **Krzysztof Lasota** – for biographies, see this issue, p. 14.