Using Security Logs for Collecting and Reporting Technical Security Metrics

Risto Vaarandi and Mauno Pihelgas

This paper has been accepted for publication at the 2014 IEEE Military Communications Conference, and the final version of the paper is included in *Proceedings of the 2014 IEEE Military Communications Conference* (DOI: 10.1109/MILCOM.2014.53)

Using Security Logs for Collecting and Reporting Technical Security Metrics

Risto Vaarandi and Mauno Pihelgas
NATO Cooperative Cyber Defence Centre of Excellence
Tallinn, Estonia
firstname.lastname@ccdcoe.org

***Abstract*—During recent years, establishing proper metrics for measuring system security has received increasing attention. Security logs contain vast amounts of information which are essential for creating many security metrics. Unfortunately, security logs are known to be very large, making their analysis a difficult task. Furthermore, recent security metrics research has focused on generic concepts, and the issue of collecting security metrics with log analysis methods has not been well studied. In this paper, we will first focus on using log analysis techniques for collecting technical security metrics from security logs of common types (e.g., network IDS alarm logs, workstation logs, and Netflow data sets). We will also describe a production framework for collecting and reporting technical security metrics which is based on novel open-source technologies for big data.**

Keywords—security metrics; security log analysis

I. INTRODUCTION

During recent years, the question "how to measure system security?" has received increasing attention, and has been addressed in a number of academic papers [1–6], governmental research reports [7–9], standards [10], books [11–12], and various other documents like recommendations and domain overviews [13–15]. For easing the process of measuring the system security, the notion of *security metric* is employed by most researchers and practitioners. Although this notion is defined in a slightly different way in different documents, all sources agree that security metric refers to a standard of measurement. For example, one can define the metric *number of port scanners per day* which involves collecting relevant data from a firewall after the end of each day. In several sources, the following common properties of a good security metric have been identified [4, 9, 11–13]:

- It is unambiguous and meaningful for predefined purposes, making sense to the human analyst
- Taking measurements for the metric should not involve significant cost impact
- Measurements should be taken consistently using the same methodology, with appropriate time frames between measurements, and preferably through an automated data collection procedure

It is often hard to find a metric that is equally meaningful for every possible audience. For example, while the metric *number of port scanners per day* is useful for security administrators, it has little value to a higher level executive who is interested in business level metrics. In a recent study [14], security metrics are classified by their expected audience into *management*, *operational*, and *technical* metrics. Technical metrics provide details for security experts, but also a foundation for other two metric classes which are primarily meant for different levels of management [14].

Even if the metric is meaningful for a particular audience, the knowledge of a wider context around the metric is often useful for increasing its clarity [13]. For instance, the metric *number of port scanners per day* does not make much sense if one is looking only at a single measurement taken for the last 24 hours, since it is not known what the usual values for this metric in a given environment are. Furthermore, during the metrics collection process the knowledge about the surrounding environment should be used. For example, if known false positive alarms are excluded when the metric *number of network attacks per day* is collected, the value of this metric will greatly increase.

Although metrics related issues have been studied in a number of sources, they often lack detailed recommendations for implementing security metrics collection and reporting system. Furthermore, since many metrics can only be obtained from security logs that are often very large in size, metrics collection requires a log management solution for big data with efficient searching and reporting capabilities. However, many traditional log management solutions are not able to cope with big data which creates a serious obstacle for metrics collection. Moreover, high-end commercial solutions are not affordable for many smaller institutions.

Also, in existing literature metrics reporting is often seen as the generation of static reports to end users. One notable exception is a hierarchical visualization architecture proposed by Savola and Heinonen [3] which supports interactive navigation from generic metrics to underlying more specific metrics. We take a step further and argue that the security metrics reporting system should be able to access raw security data sets (such as security logs) and have efficient drill-down functionality – the generation of more specific reports on user-defined queries, and the identification of individual entities in raw security data (such as log messages or Netflow records). This allows the human analyst to study the details behind the

metric and increase its meaningfulness [3], and also helps to find root causes for anomalies which have been spotted in reported metric values.

During the last few years, open-source technologies for storing, searching, analyzing, and visualizing very large log data sets have rapidly emerged (e.g., Elasticsearch, Kibana, and Graylog2). These technologies can be used for creating a cost-efficient security metrics collection and reporting system with dynamic reporting and drill-down capabilities for security logs.

Unfortunately, these developments have received little attention in recent academic and industrial papers, and previous works have not focused on using security logs for metrics collection. This paper addresses this research gap, and discusses log analysis methods and open-source solutions for collecting and reporting technical security metrics. The remainder of the paper is organized as follows – section II provides an overview of related work, section III discusses log analysis methods for extracting metrics from security logs of common types, section IV describes an open-source based production framework for security metrics collection and reporting, and section V concludes the paper.

II. Related Work

One of the earliest works which suggested the use of security metrics was a book by Jaquith [11]. The book describes the properties of a good metric and provides a detailed discussion on how to report metrics. During the past few years, the use of security metrics has been proposed for a wide variety of domains, including SCADA and other control systems [5, 7, 8], cloud computing [6], application security [2], software design [1], and assessment of cyber threats [9].

Recently, the Center for Internet Security has published a report [14] on standard metric and data definitions that can be used across different organizations, in order to collect and analyze data on security processes and outcomes. The report offers universal guidelines on implementing security metrics program in an organization. It also proposes 28 metric definitions that have all been categorized in two ways, either by relevant business function or by purpose and target audience. These metrics are meant to serve as a starting point for organizations which are beginning to implement their metrics program. Nevertheless, the report does not offer any detailed recommendations for implementing a production system for metrics collection and reporting.

A recent paper by the Council on CyberSecurity [15] describes 20 critical controls for achieving effective cyber defense. The paper considers security log collection and analysis as one of the critical controls, and also emphasizes the importance of IDS and Netflow based network monitoring. Although the main focus of the paper is not on security metrics, it proposes a number of specific metrics for measuring the efficiency of suggested cyber defense controls.

Security metrics have also been discussed in the ISO/IEC 27004:2009 standard [10] which aims to measure, report on, and systematically improve the effectiveness of Information Security Management Systems that have been specified in ISO/IEC 27001. However, the current standard has been criticized by some security practitioners for being too generic and lacking practical guidance on which particular metrics to collect [12].

In addition to aforementioned sources, a recent book by Brotby and Hinson [12] offers practical recommendations and examples on implementing security metrics program. The authors of the book propose the novel PRAGMATIC methodology for defining, scoring, and ranking metrics, in order to identify the most beneficial ones for different audiences (e.g., security professionals, managers, and other stakeholders). Furthermore, the book describes over 150 example metrics, in order to help the reader to build his/her own metrics program.

Apart from generic studies, security metrics have also been suggested for measuring specific aspects of cyber security. For example, a recent study conducted in Sandia National Labs [9] discusses possible metrics for cyber threats (malicious organizations and individuals), and proposes the use of the threat matrix model for assessing cyber threats.

III. Extracting Technical Security Metrics from Security Logs

As discussed in the previous section, existing works often focus on generic security metric concepts, and lack recommendations for implementing production systems for metrics collection and reporting. In this section, we will discuss the use of log analysis methods and tools for several common security log types, in order to collect technical security metrics.

A. Extracting Security Metrics from Network IDS Alarm Logs

Today, network IDSs are used by vast majority of institutions which are processing data of critical importance. Therefore, IDS alarm based security metrics are a popular choice for measuring the system security and the threat level against the local network. In production systems, it is a common practice to measure the number of IDS alarms per hour, day, or some other time frame, and report this metric as time-series data to the human analyst. Based on IDS alarm attributes, a number of additional metrics can be defined (e.g., the number of botnet related alarms per time frame). Also, it is often worthwhile to use event correlation for detecting alarm patterns that correspond to specific attacks, since this allows for creating metrics for these attacks (section IIIc provides a detailed example on how to employ event correlation for extracting metrics from log data).

Although network IDS alarm based metrics are commonly used, they are sensitive to false positives, especially if a larger volume of false positive alarms appears and the reported metric becomes seriously distorted. Furthermore, IDS signatures which detect frequent bad traffic of low importance (such as probes from well-known Internet worms) can routinely trigger many alarms over longer periods of time [16]. Such alarms form the background noise that might again distort reported metrics. Although filters for known false positives and threats of low importance can be created manually, new types of false positives and noise might be easily introduced with signature updates and changes in the environment. In order to alleviate this problem, we have proposed a real-time IDS alarm

classification algorithm during our past research which is able to distinguish false positives and noise from interesting alarms [16]. The proposed algorithm applies various data mining techniques to past IDS alarm logs, in order to learn patterns that describe noise and false positive alarms, and uses detected patterns for real-time IDS alarm classification. The learning step of the algorithm is periodically repeated (e.g., once in 24 hours), in order update the classification knowledge and adjust to changes in the surrounding environment. While our previous paper described preliminary results of using this algorithm [16], we have employed this method for several years in production. One of the purposes for introducing this method was to filter out irrelevant IDS alarms and calculate more meaningful security metrics from important alarms only.

B. Extracting Security Metrics from Netflow Data

Netflow is a network traffic statistics collection protocol developed by Cisco Systems. If a network device has Netflow statistics collection enabled, it will extract data from the header of each observed packet, and store these data in Netflow records. For each network flow a separate record is maintained, where the network flow is identified by the transport protocol, source and destination IP addresses, source and destination port numbers, and couple of other fields (such as Type of Service). Apart from transport protocol, source and destination transport addresses, each Netflow record contains a number of additional fields, including the total number of packets and bytes for the network flow, the union of all TCP flags seen in the packet headers of the flow, and the start and end times of the flow. Collecting Netflow data in Internet backbone networks requires a lot of resources, and is often accomplished with sampling in such environments (e.g., only 0.01% of the packets are processed). However, in institutional networks that handle much less traffic than Internet backbones, collection without sampling is often feasible. This provides a detailed picture of all communications in monitored network segments without storing full packet payloads. In the following discussion, we assume that Netflow data are collected without sampling.

When Netflow data are collected, they can be used for calculating a number of security metrics. Firstly, it is often useful to set up blacklist-based security metrics, in order to monitor and collect trend information on data exchange with known malicious, compromised, or suspicious peers in the Internet. Some security institutions such as EmergingThreats are maintaining publicly available blacklists of known bad hosts, including botnet C&C nodes, compromised nodes, and Tor nodes (e.g., see [17]). When these blacklists are frequently downloaded and used for querying collected Netflow data sets, it is straightforward to create metrics that describe communications with malicious peers. For example, Fig. 1 and Fig. 2 depict metrics which reflect daily traffic exchanged with known compromised nodes and Tor network nodes during the last 2 months (Fig. 1 and Fig. 2 display the average bits-per-second traffic rate for each day, and data for the metrics have been collected on the outer network perimeter of a large institution).

Collected Netflow data can also be used for creating metrics for abnormal and potentially malicious network activity which supplement similar IDS alarm based metrics. For example, since a Netflow record contains a field for holding the union of all observed TCP flags for the given flow, it is straightforward to write a filtering condition for detecting flows with illegal flag combinations (e.g., TCP FIN flag never appears without TCP ACK flag in normal network traffic). Based on detected flows, metrics can be set up that describe various aspects of illegal traffic (such as the number of distinct Internet hosts per hour which are sources of abnormal traffic).

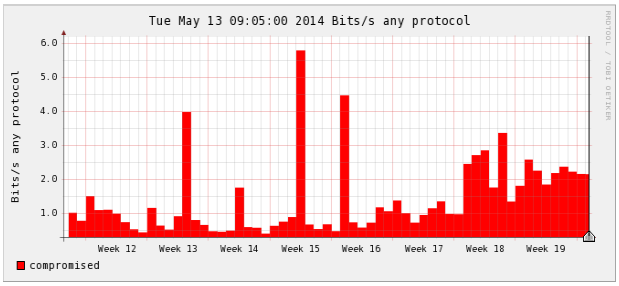

Fig. 1. Daily traffic exchanged with known compromised hosts (reflects probing activity from infected Internet hosts, but also suspicious or unwanted communications from local network to malicious hosts)

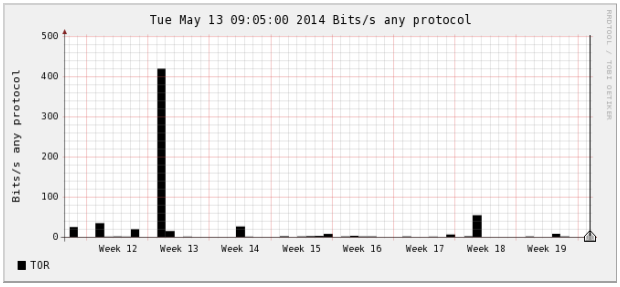

Fig. 2. Daily traffic exchanged with known Tor network hosts (reflects Tor client traffic to institutional web site and other public services, but can also reveal Tor clients in the institutional network)

Netflow statistics collection and analysis can also be employed in private networks, in order to discover illegal devices and services, malicious insiders, and infected computers, since they often manifest themselves through anomalous network traffic which differs from regular network usage patterns. Netflow based network monitoring offers some unique advantages. Firstly, it does not involve packet payload inspection and consumes much less computing resources than network IDS. Also, traditionally illegal devices and services are detected by scanning the entire network with dedicated tools. However, this is an expensive and time consuming procedure which might also alert the owner of illegal device or service. In contrast, Netflow based detection is stealthy and does not consume network bandwidth.

However, service detection from Netflow data involves several caveats. Most notably, due to commonly found design flaws in Netflow implementations, some Netflow records can have imprecise timestamps [18, 19]. As a result, if the Netflow record for service-to-client traffic is incorrectly tagged with an earlier timestamp than the record for client-to-service traffic, the service can be mistakenly taken for the client.

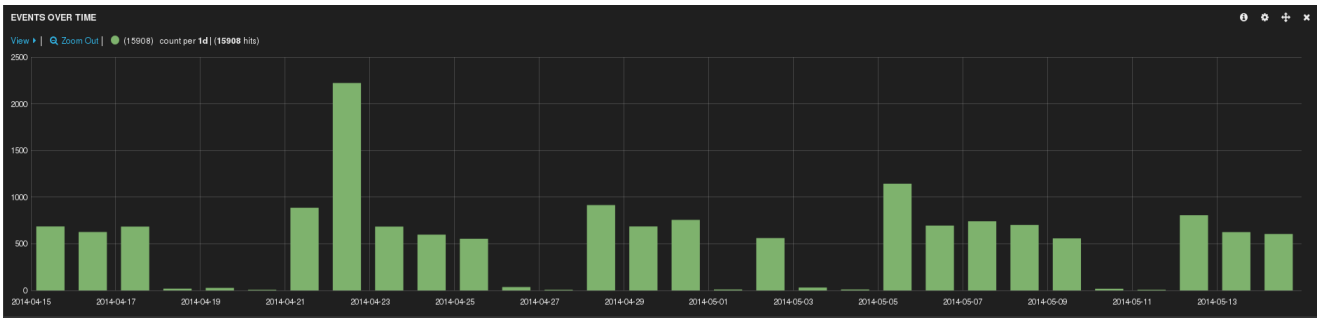

Fig. 3. Daily login failures for all institutional Windows workstations (note that unexpected spike in April 22 reflects an account probing activity by malware which infected one of the workstations, but was promptly detected and removed)

Also, some router-based Netflow implementations might leave the "union-of-flags" field unset [18], and this exacerbates service detection further. For addressing these issues, various heuristic techniques have been suggested [19, 20]. After services and hosts have been identified from Netflow data and compared with the lists of legitimate hosts and services, it is straightforward to identify illegal devices and services, and create corresponding security metrics (e.g., the number of illegal devices by organizational unit as recommended in [15]).

In order to detect anomalous network traffic in private networks that might indicate malware infection, illegal data access, or malicious insider activity, various methods can be applied to Netflow data sets. For example, if workstations in the private network are using a small set of well-known services, a simple filtering condition might be written which reports workstation traffic not related to known services (this would easily allow to find a number of network misuse cases, such as malware propagation from an infected workstation to other workstations). For more complex networks, automated methods might be used that learn normal network usage patterns from past Netflow data sets, and use detected knowledge for finding deviations from normal behavior. For example, during our past research, we have developed a method which learns and updates service usage profiles for each individual client node and the entire network, and uses these profiles for real-time detection of anomalous TCP and UDP network flows [20]. Once anomalous network flows have been identified, it is straightforward to create metrics from them (e.g., the number of anomalous TCP flows per 24 hours).

C. *Extracting Security Metrics from Workstation Logs*

Workstations in institutional networks are major targets for malware and targeted attacks, and therefore their monitoring and the creation of security metrics from monitoring information plays an important role. Significant amount of workstation security status information can be obtained from workstation logs, such as login failures into the workstation, antivirus alerts, etc. Unfortunately, since workstations create large volumes of log data, the collection and analysis of these data is expensive and thus often neglected. A recent SANS paper by Anthony [21] suggests several strategies for setting up a centralized log collection and analysis framework for Windows workstations, and identifies a number of event types which should be collected, monitored, and correlated. In order to minimize the cost of log collection and analysis, the paper [21] proposes to send events of relevant types only to the central collection point where they are analyzed with SEC [22]. Another recent paper [23] provides a number of detailed recommendations for monitoring Windows event logs, in order to detect adversarial activities.

One group of well-known security event types in the workstation log reflects login attempts into the local workstation. Note that these event types are not Windows-specific, but can also be easily identified for UNIX-like workstation platforms (e.g., login failures for SSH or FTP services). As discussed in [15], user account monitoring and control is one of the critical cyber defense controls. Also, the monitoring of unusual login attempts helps to detect malware propagation [21] and malicious insiders [15]. For these reasons, it makes sense to set up metrics that describe different types of successful and failed login attempts into workstations (e.g., the number of successful logins from unexpected remote hosts per 1 hour). For example, Fig. 3 depicts a metric which presents daily numbers of login failures for all institutional workstations during 1 month time frame (this example metric has been collected in a large institutional network from the logs of thousands of workstations). In addition to the above scenario, several other metrics could be collected from workstation logs, for example, the number of distinct workstations or accounts with login failures in a given timeframe (sudden increase in the number of hosts and accounts might indicate massive account probing over the entire network, in order to get unauthorized access to data).

Also, event correlation techniques are often useful for creating more meaningful metrics from workstation log events. For example, instead of including each accidental login failure in a relevant metric, the login failure might only be taken into account if it is not followed by a successful login within a reasonable amount of time (e.g., 60 seconds). Fig. 4 displays an example SEC event correlation rule for Linux platform which processes events for failed and successful SSH login attempts, and sends collected metric values to Graphite reporting and visualization system.

Apart from event types mentioned above, workstation logs contain a wide variety of other events that can be harnessed for creating useful security metrics. For example, Fig. 5 presents two metrics which indicate daily numbers of update and patching failures for Windows operating system and Internet Explorer (depicted metrics have been collected from the logs of thousands of Windows workstations of a large institution, and

the metrics are used for measuring the quality of the patching process). Finally, it should be noted that the relevance of a particular event type for the metric collection process depends on the nature of the environment (e.g., in many regular networks USB insertion events are unimportant, while in classified networks they often deserve closer inspection).

```
# if the login failure is not followed by a successful login
# within 60 seconds, include the failure in the metric

type=PairWithWindow
ptype=RegExp
pattern=sshd\[\d+\]: Failed .+ for (?:invalid user )?(\S+) \
from ([\d.]+) port \d+ ssh2
desc=SSH login failed for user $1 from IP $2
action=lcall %count %count -> ( sub { ++$_[0] } )
ptype2=RegExp
pattern2=sshd\[\d+\]: Accepted .+ for $1 from $2 port \d+ ssh2
desc2=SSH login successful for user %1 from IP %2
action2=none
window=60

# send the current metric value to the Graphite server
# in every 5 minutes and reset the metric counter

type=Calendar
time=*/5 * * * *
desc=report SSH login failure metric once in 5 minutes
action=if %count () else ( assign %count 0 ); eval %n "\n"; \
  tcpsock graphite:2003 login.failures.ssh.total5m %count %u%n; \
  free %count
```

Fig. 4. Sample SEC ruleset for collecting the metric *number of SSH login failures for all workstations per 5 minutes*, and sending it to Graphite reporting and visualization platform

D. Extracting Security Metrics from Other Logs

Security metrics can be extracted from a number of other logs, including server and router logs, firewall logs, service logs, etc. Establishing proper metrics is especially important for public services that can be accessed from the Internet. Apart from creating metrics from events which describe known security issues, one can also process both regular and unusual events. For example, while normally most HTTP client requests are for existing web pages, occasionally clients might request non-existing or forbidden pages that produce HTTP log entries with 4xx response codes. However, unexpectedly large volumes of 4xx log messages or normal 200 messages can indicate a reconnaissance scan or the start of a DDoS attack. Deriving metrics from such messages will help to assess the threat level for the service. Also, applying event correlation techniques for a service log or cross-correlating messages from different logs (e.g., service log and IDS log) is often useful for detecting advanced threats, and creating metrics for these threats (see our previous papers [22, 24] for examples on how to employ SEC for various event correlation tasks).

IV. Security Metrics Collection and Reporting Framework for Security Logs

In this section, we will describe a production framework for collecting and reporting security metrics that harnesses log analysis techniques outlined in the previous section. The framework has been set up in a large institution which is an important part of the national critical information infrastructure, and has a complex organizational network consisting of many thousands of workstations, servers, network devices, IDSs, firewalls, and other nodes.

As discussed in section I, the collection of security metrics should not involve significant cost impact, and it should be preferably done with an automated collection system. In order to address these requirements, our framework is centralized, since analyzing security logs locally at workstations, servers, and other nodes would impose additional load on them, and interfere with normal system activities. Also, decentralized log analysis would considerably increase the complexity of the metrics collection system. For reducing the implementation costs, our framework runs on CentOS Linux platform and is based on open-source solutions.

Our centralized framework is receiving security log data from all relevant nodes in the organizational network over the *syslog* and Netflow protocols. For events which are not natively in *syslog* format, appropriate format conversion gateways are used (e.g., Windows EventLog messages are converted to *syslog* format with Nxlog [25]). Incoming *syslog* events are received by several central log collection servers that are running Syslog-ng [26], and collected events are further correlated by a number of SEC instances. During the event correlation, a number of security metrics are extracted and sent to Graphite reporting and visualization system. Graphite [27] has been specifically designed for performing computations on time-series data, and generating wide variety of graphs and reports from computation results.

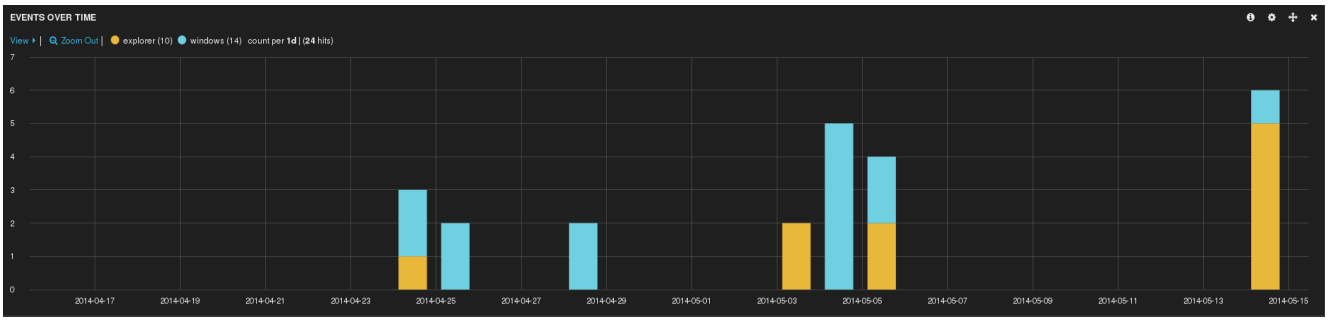

Fig. 5. Daily updating and patching failures of Windows operating system and Internet Explorer for all institutional Windows workstations

For collecting unsampled Netflow data from network devices and dedicated probes, our centralized framework uses NfSen [28] that is a flexible Netflow visualization tool with drill-down capabilities (see Fig. 1 and Fig. 2 for example metric reports generated with NfSen).

From *syslog* and Netflow collection servers, all *syslog* events and Netflow data are forwarded to a central Elasticsearch [29] database cluster which the end users are accessing through Kibana visualization interface [30]. The database cluster runs on several CentOS Linux nodes with two quad-core CPUs and 48GB of memory. Currently, almost 100 million security log records are stored in Elasticsearch on a daily basis (the records are kept in Elasticsearch for three months). In order to receive, parse, and store these data, we are using Rsyslog [31] and Logstash [32]. According to our past experiments, Rsyslog is one of the most efficient *syslog* servers with Elasticsearch support, while Logstash supports flexible parsing of *syslog* and Netflow data [33]. In Kibana, more than 20 dashboards have been set up for displaying various security metrics (Fig. 3 and Fig. 5 display two metric report examples). All reports generated with Kibana are interactive and allow for drilling down to more specific reports and individual log records. Therefore, after spotting an anomaly in a metric report, the root cause events for this anomaly can be quickly identified.

V. Conclusion and Future Work

In this paper, we have discussed the use of security logs for collecting and reporting of security metrics, and have reviewed a number of metrics collection scenarios for common security log types. Also, we have described a production framework for metrics collection and reporting which is based on open-source log management technologies. For the future work, we plan to research log analysis methods for insider threat detection, and to implement relevant algorithms within our framework.

References

[1] B. Alshammari, C. Fidge, and D. Corney, "Security Metrics for Object-Oriented Class Designs," in *Proceedings of 2009 International Conference on Quality Software*, pp. 11-20

[2] T. Heyman, R. Scandariato, C. Huygens, and W. Joosen, "Using security patterns to combine security metrics," in *Proceedings of 2008 International Conference on Availability, Reliability and Security*, pp. 1156-1163

[3] R. M. Savola and P. Heinonen, "A Visualization and Modeling Tool for Security Metrics and Measurements Management," in *Proceedings of 2011 Information Security for South Africa Conference*, pp. 1-8

[4] R. Barabanov, S. Kowalski, and L. Yngström, "Information Security Metrics: Research Directions," University of Stockholm, Technical Report, 2011

[5] W. Boyer and M. McQueen, "Ideal Based Cyber Security Technical Metrics for Control Systems," in *Proceedings of 2007 International Conference on Critical Information Infrastructures Security*, pp. 246-260

[6] C. A. da Silva, A. S. Ferreira, and P. L. de Geus, "A Methodology for Management of Cloud Computing using Security Criteria," in *Proceedings of 2012 IEEE Latin American Conference on Cloud Computing*, pp. 49-54

[7] R. A. Kisner, W. W. Manges, L. P. MacIntyre, J. J. Nutaro, J. K. Munro, P. D. Ewing, M. Howlander, P. T. Kuruganti, R. M. Wallace, and M. M. Olama, "Cybersecurity through Real-Time Distributed Control Systems," Oak Ridge National Laboratory, Technical Report ORNL/TM-2010/30, February 2010

[8] A. McIntyre, B. Becker, and R. Halbgewachs, "Security Metrics for Process Control Systems," Sandia National Laboratories, Sandia Report SAND2007-2070P, September 2007

[9] M. Mateski, C. M. Trevino, C. K. Veitch, J. Michalski, J. M. Harris, S. Maruoka, and J. Frye, "Cyber Threat Metrics," Sandia National Laboratories, Sandia Report SAND2012-2427, March 2012

[10] ISO/IEC 27004:2009 standard "Information technology -- Security techniques -- Information security management -- Measurement", 2009

[11] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison-Wesley, 2007

[12] W. K. Brotby and G. Hinson, *PRAGMATIC Security Metrics: Applying Metametrics to Information Security*. Auerbach Publications, 2013

[13] P. E. Black, K. Scarfone, and M. Souppaya, "Cyber Security Metrics and Measures," in *Wiley Handbook of Science and Technology for Homeland Security*, John Wiley and Sons, 2009

[14] "The CIS Security Metrics," The Center for Internet Security, Technical Report, version 1.1.0, November 1 2010

[15] "The Critical Controls for Effective Cyber Defense," Council on CyberSecurity, Technical Report, version 5.0, 2014

[16] R. Vaarandi and K. Podiņš, "Network IDS Alert Classification with Frequent Itemset Mining and Data Clustering," in *Proceedings of the 2010 IEEE Conference on Network and Service Management*, pp. 451-456

[17] http://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt

[18] R. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras, "Measurement Artifacts in NetFlow Data," in *Proceedings of the 2013 Passive and Active Measurement Conference*, pp. 1-10

[19] B. Trammell, B. Tellenbach, D. Schatzmann, and M. Burkhardt, "Peeling Away Timing Error in NetFlow Data," in *Proceedings of the 2011 Passive and Active Measurement Conference*, pp. 194-203

[20] R. Vaarandi, "Detecting Anomalous Network Traffic in Organizational Private Networks," in *Proceedings of the 2013 IEEE CogSIMA Conference*, pp. 285-292

[21] R. Anthony, "Detecting Security Incidents Using Windows Workstation Event Logs," SANS Institute, InfoSec Reading Room Paper, June 19 2013

[22] R. Vaarandi, "Simple Event Correlator for real-time security log monitoring," Hakin9 Magazine, vol. 1/2006 (6), pp. 28-39, 2006

[23] "Spotting the Adversary with Windows Event Log Monitoring," National Security Agency/Central Security Service, Information Assurance Directorate, Technical Report, Revision 2, December 16 2013

[24] R. Vaarandi and M. R. Grimaila, "Security Event Processing with Simple Event Correlator," Information Systems Security Association Journal, vol. 10(8), pp. 30-37, 2012

[25] http://nxlog.org

[26] http://www.balabit.com/network-security/syslog-ng

[27] http://graphite.readthedocs.org

[28] http://nfsen.sourceforge.net

[29] http://www.elasticsearch.org

[30] http://www.elasticsearch.org/overview/kibana/

[31] http://www.rsyslog.com

[32] http://logstash.net

[33] R. Vaarandi and P. Niziński, "Comparative Analysis of Open-Source Log Management Solutions for Security Monitoring and Network Forensics," in *Proceedings of the 2013 European Conference on Information Warfare and Security*, pp. 278-287