Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 137 (2018) 109–119

Procedia Computer Science

www.elsevier.com/locate/procedia

SEMANTiCS 2018 – 14th International Conference on Semantic Systems

Taming the logs – Vocabularies for semantic security analysis

Andreas Ekelhart[a,*], Elmar Kiesling[b], Kabul Kurniawan[b]

[a]SBA Research, Favoritenstrasse 16, Vienna 1040, Austria
[b]TU Wien, Favoritenstrasse 9-11, Vienna 1040, Austria

Abstract

Due to the growing complexity of information systems and the increasing prevalence and sophistication of threats, security management has become an enormously challenging task. To identify suspicious activities, security analysts need to monitor their systems constantly, which involves coping with high volumes of heterogeneous log data from various sources. Processes to aggregate these disparate logs and trigger alerts when particular events occur are often automated today. However, these methods are typically based on regular expressions and statistical correlations and do not involve any interpretation of the context in which an event occurred and do not allow for inference or sophisticated rules. Inspection and in-depth analysis of log information to link events from various sources (e.g., firewall, syslog, web server log, database log) and establish causal chains has therefore largely remained a tedious manual search process that scales poorly with a growing number of heterogeneous log sources, log volumes, and the increasing complexity of attacks.

In this paper, we make the case for a semantic approach to tackle these challenges. By lifting raw log data and modeling their context, events can be linked to rich background knowledge, integrated based on causal relations, and interpreted in a context-specific manner. This builds a foundation for more comprehensive extraction of the meaning of events from unstructured log messages. Based on the results, we envision a platform to partly automate security monitoring and support analysts in coping with fast evolving threat landscapes, alleviate alert fatigue, improve situational awareness, and expedite incidence response.

Peer-review under responsibility of the scientific committee of the SEMANTiCS 2018 – 14th International Conference on Semantic Systems.

Keywords: semantic extraction; log vocabularies; log analysis; security analysis

1. Introduction

According to industry analyses, today's ICT systems are threatened on an unprecedented scale by increasingly sophisticated and targeted attacks [31, 15]. This has resulted in a series of widely publicized data breach cases [16] associated with enormous economic costs [11]. A key issue in this context is the lack of awareness about sophisticated multi-vector attacks, which are typically difficult to detect automatically using standard intrusion detection methods. This

* Corresponding author. Tel.: +43-1-505-3688; fax: +43-1-505-8888.
E-mail address: andreas.ekelhart@sba-research.org

1877-0509
Peer-review under responsibility of the scientific committee of the SEMANTiCS 2018 – 14th International Conference on Semantic Systems.
10.1016/j.procs.2018.09.011

slow detection and limited understanding of the scope of security incidents severely inhibits organizations' ability to react adequately and take timely measures to mitigate and contain their impact. According to an industry survey, 59% of companies do not have adequate intelligence or are unsure about attempted attacks and their impact. Furthermore, 51% say their security solutions do not inform them or they are unsure if their solution can inform them about the root causes of an attack. [25]

These difficulties are not necessarily caused by a lack of data, as most hard- and software components provide comprehensive logging facilities that produce fine-grained and high-frequency information about their state and about observed events. Consequently, organizations' failure to detect and respond to security incidents is not caused by a lack of clues that point to attacks, but by the rapid growth of log data and the manual analysis, which is becoming less and less feasible.

Apart from sheer volume, this is further complicated by the fact that these logs are typically weakly structured, use a variety of inconsistent formats and terminologies, and are spread across different files and disparate systems. To detect sophisticated multi-vector attacks, it is necessary to identify related events and connect them to create a complete picture for the identification of malicious activity. Isolated indicators are often inconspicuous in their local context and it is therefore necessary to harmonize and integrate disparate log information to obtain a complete picture. Furthermore, the interpretation of log information is highly context-specific, which makes it difficult for monitoring applications to identify relevant information without a deeper understanding of their context. Manual log analysis by human experts as a last resort does not scale in this context and there is a lack of automated mechanisms to integrate and interpret security information. Consequently, security analysts struggle to cope with the enormous volume of raw log data, to extract insights from these heterogeneous sources, and to identify and respond to increasingly complex attacks.

In this paper, we make the case for a semantic approach towards security analysis. This approach has the potential to reconcile today's highly fragmented log information landscape by extracting and interlinking relevant security information, facilitating automated inference, and providing security analysts with an integrated perspective that promotes understanding and improves situational awareness.

To this end, we develop a set of vocabularies for the uniform representation of log events and discuss how a solid conceptual foundation facilitates linking of disparate log information, extraction of relevant events, contextualization, and enrichment with background knowledge. Furthermore, we describe an architecture for semantic log processing that provides integrated access to log information via various interfaces. This creates a foundation for semantic security monitoring, incidence response, and forensic applications. In particular, the proposed approach will enable context-aware decision support and thereby overcome the limited scope and lack of interpretation capabilities of current security monitoring and response technologies such as virus scanners, intrusion detection systems (IDSs), and security incident and event management (SIEM) systems. Furthermore, the developed vocabularies provide a foundation for sharing threat intelligence across organizations.

The remainder of this paper is structured as follows: Section 2 introduces the wider context of the SEPSES semantic log analysis architecture (Section 2.1) and then specifically focuses on log extraction (Section 2.2) and vocabularies (Section 2.3); Section 3 discusses results from our prototypical implementation and illustrates how the approach addresses current challenges in security monitoring; Section 4 provides an overview of related work within industry and academia; Section 5 closes with conclusions and an outlook on future work.

2. Semantic security log analysis

The log extraction approach introduced in this paper constitutes the core of a larger semantic log analysis framework called SEPSES[1], which will provide a platform for semantic security monitoring, auditing, and forensics.

2.1. SEPSES Architecture

The SEPSES architecture is designed to facilitate scalable semantic processing, integration, aggregation, and interpretation of heterogeneous logs in highly diverse environments. Its main components are illustrated in Figure 1.

[1] Semantic Processing of Security Event Streams

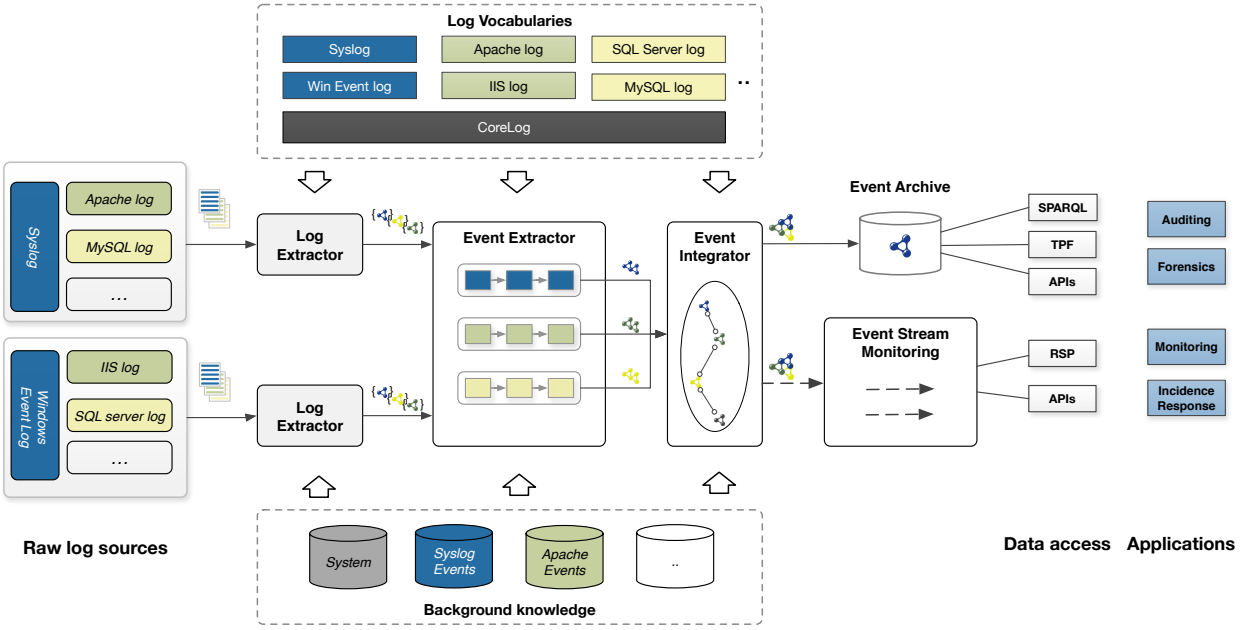

Fig. 1: Semantic log extraction architecture

In order to provide an integrated perspective on all relevant security information and enable semantic processing, it is first necessary to collect log data from a multitude of *log sources* (e.g., system logs, network logs, application logs, etc.) in different formats. A wide range of tools to acquire and store log information locally exists, but to provide an integrated perspective it is necessary to transfer local log information to a central repository and harmonize the heterogeneous log data. This typically involves local log agents (installed on local machines) which harvest log information from various sources and ship them to extractor components. This approach is flexible and allows for scalable log extraction at an appropriate level of centralization. The options range from fully decentralized extraction on the local machines, to fully centralized and load-balanced extraction on servers.

In our architecture, the *log extractor* components then transform the raw log data, received from multiple sources in various formats, into JSON-LD [13], a lightweight Linked Data format. Because most modern log management systems use JSON to encode and transport log messages, a key advantage of this approach is that JSON-LD annotations can be added easily in an efficient and scalable manner in order to make the data interoperable at web scale. This transformation from raw log data into an integrated JSON-LD log stream applies a set of well-specified log vocabularies and thereby solves syntactic interoperability challenges and – through alignment of formats and scales – creates a minimum level of semantic interoperability (cf. Section 2.2). To this end, the log extractors make use of a modular *log vocabulary* stack – described in Section 2.3 – that consists of `slog:core`, a foundation that provides basic terms to describe log messages irrespective of their source, and specialized vocabularies that extend the core vocabulary with log source-specific terms.

The harmonized stream of RDFized log messages in JSON-LD format is then forwarded to *event extraction pipelines*, which apply a sequence of enrichment, alignment, reconciliation, and integration steps to extract an interpretation of the meaning contained within the log messages[2]. These pipelines can combine a variety of information extraction techniques (e.g., named entity recognition, classifiers, and hand-written regular expressions) with inference, linking, and rule-based approaches. In summary, extraction pipelines link incoming log messages to rich *background knowledge* and obtain context-sensitive, machine-readable interpretations of security-related events.

Subsequently, extracted log events are sent to the *event integration* component, which establishes links between related events, irrespective of their original log source (e.g., combining the traces of a remote login event from the

[2] Out of the scope of the present paper

client, the server, and the firewall) and thereby establishes causal relations between isolated low-level events and integrates them into high-level events. For instance, a single high-level *login* event can be associated with multiple low-level events, i.e., a series of syslog events, authentication events, firewall events, etc. This step can be supported by statistical, learning-based, and (stream) reasoning-based event integration approaches.

Finally, the integrated event stream can be monitored with *RDF stream processing* techniques and forwarded to an *RDF log archive*, i.e., a triple store, for later analysis. The prepared log data can be made available for log auditing, analysis, and forensic applications via various interfaces. These interfaces, including APIs, SPARQL, and Linked Data Fragments (LDF) interfaces, constitute the data access layer of the architecture. One option for (near) real-time log monitoring by applications is to register continuous SPARQL queries.

In the following section, we focus specifically on the log extraction of the proposed architecture.

2.2. Log extraction

In our prototypical implementation, we use Logstash [32], an open source tool for collecting and parsing logs, as *log extraction* component. Furthermore, we use Filebeat to ship raw log data from remote agents installed on local systems to the log extractor.

To inject JSON-LD annotations into the JSON log stream, we configured Logstash with custom filters that restructure the data to conform to the SEPSES log vocabularies (Section 2.3) and add appropriate `@context` and `@id` annotations to yield valid JSON-LD output. A key benefit of this approach over plugin based configuration, hard-coded extraction or the use of generic RDF mapping tools (e.g., RML [7]) is that it can perform initial lifting from many sources in a flexible and scalable manner using tools that are optimized for large-scale, high-throughput log processing. Furthermore, it permits (near) real-time extraction and usage of the lifted log streams in stream processing scenarios.

The log vocabularies and JSON-LD format provide a well-defined interface that makes it possible to exchange the log extraction component (i.e., Logstash in our implementation) or combine it with other extraction tools, provided they can be adapted to produce JSON-LD output.

Whereas the interpretation of the log messages will be handled in the *event extraction* phase, the log extraction process does perform preliminary harmonization steps and results in an integrated, uniform representation of log streams. For instance, log parsing and transformation already harmonizes heterogeneous time stamp formats and represents them uniformly as `xsd:dataTime` properties.

Furthermore, the model of the RDF log streams is structured in a way that provides ”attachment points” for subsequent enrichment, alignment, entity reconciliation, and linking of the log information to background knowledge in the event extraction and integration phases. These attachment points are modeled as auxiliary nodes that have a number of literal properties and a randomly generated identifier (UUID). These nodes are subsequently used for alignment (e.g., harmonizing severity levels in logs from different operating systems and hence different severity scales), entity reconciliation, and linking to background knowledge (e.g., link hosts that appear in a log stream to system background knowledge, irrespective of whether they are identified by IP, hostname, MAC address, etc.). In the event extraction and integration phase, the literal properties of these nodes will be used to identify the associated concepts and create `owl:sameAs` links between them.

2.3. Log vocabularies

The event extraction approach introduced in this paper transforms raw log events into a uniform RDF representation. This necessitates appropriate vocabularies to express the heterogeneous log information. As a foundation of all log vocabularies, the `core`[3] vocabulary provides basic concepts common for all log messages produced by a log extractor, most importantly the *LogEntry* class that represents a single generic log entry, as well as the *Host* and *Logtype* classes. The `core`log vocabulary also defines several properties, i.e., *hostName*, *ipAddress* with sub-properties *ip4Address* and *ip6Address*, *timestamp*, *logMessage*, and *logFilePath*.

[3] `https://purl.org/sepses/vocab/log/coreLog`, recommended prefix: `scl`

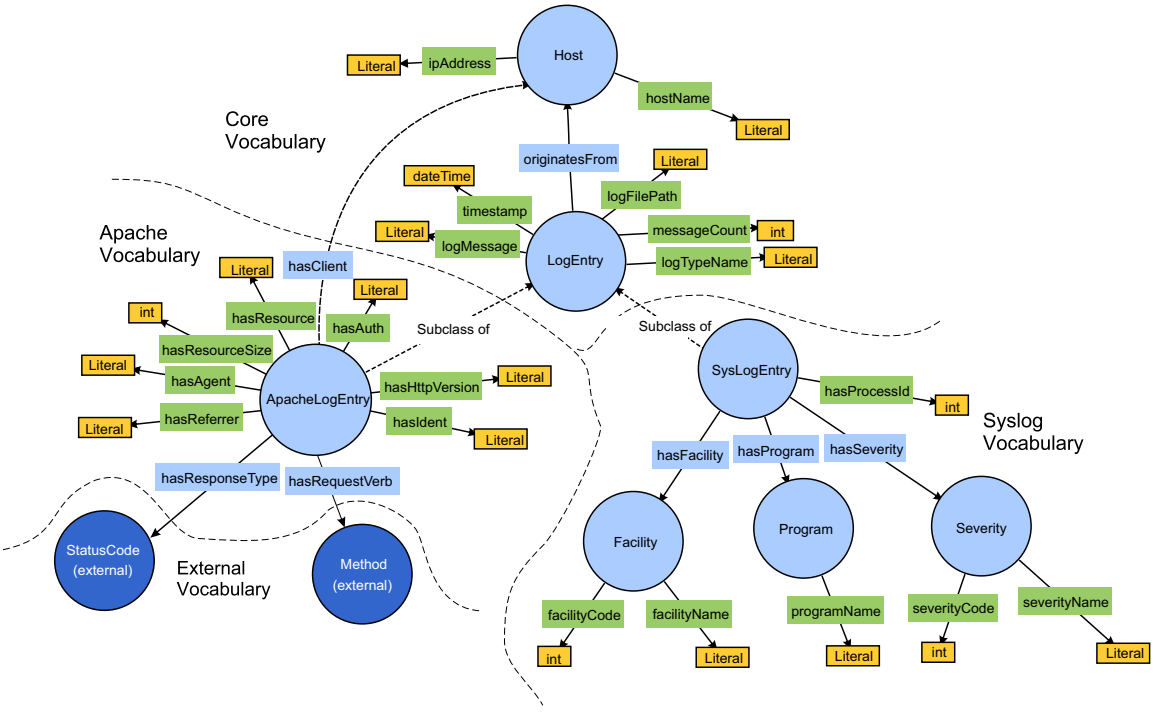

Fig. 2: Log vocabularies

A key principle in the design of our conceptual models is modularity, i.e., we organize the concepts into specialized vocabularies that can be mixed and matched, which is key to tackle log heterogeneity. This is achieved with log source-specific vocabularies that extend the *LogEntry* class with more specific subclasses. In our example illustrated in Figure 2, we use *ApacheLogEntry* provided by the *apacheLog*[4] vocabulary for Apache web server logs and *sysLogEntry* provided by *syslog*[5] vocabulary for Unix system logs.

In the design of these vocabularies, we aim to reuse existing vocabularies wherever possible. For instance, we reuse the existing `StatusCode` and `Method` concepts from the *http*[6] vocabulary for `ResponseType` and `RequestVerb`, respectively.

3. Implementation and preliminary results

To validate our approach w.r.t. current security monitoring challenges, we implemented the acquisition and extraction components (cf. Section 2.1) using Logstash as a platform for event collection and processing as well as our log vocabularies and system ontologies. We then set up a test system in a virtual machine that ran an Apache web server. From this machine, we acquired syslog and Apache log data using Filebeat and set up a custom Logstash configuration that transforms the log stream into semantically explicit JSON-LD (cf. Section 2.2). In the following, we briefly discuss current major challenges in security monitoring and illustrate with minimalist examples how our semantic log processing approach can help to tackle them.

[4] https://purl.org/sepses/vocab/log/apacheLog, recommended prefix: `apacheLog`
[5] https://purl.org/sepses/vocab/log/sysLog, recommended prefix: `syslog`
[6] http://www.w3.org/2011/http

```
Apr 9 09:37:47 kabul-VirtualBox systemd[1]: Mounted Huge Pages File System.
```

Fig. 3: Original raw log message

Volume. The ever-growing volume of data relevant for security analyses poses a significant challenge – hard- and software components produce fine-grained and high-frequency log data. Whereas our approach does not reduce the amount of generated log data itself, it has the potential to reduce the data a security analyst must consider and inspect (e.g., by harmonizing logs, filtering noise, and aggregating events). Furthermore, the semantic integration provides a foundation for powerful semantic querying that can, for instance, use inference for generalizations or apply context in the automatic interpretation of events. This makes it easier to cope with large amounts of data and we expect that this approach will be vastly more efficient than manual auditing and more effective than simpler log correlation approaches that tend to produce a lot of false positives and thereby sometimes contribute rather than solve the problem of data volume.

Heterogeneity. Logging data is typically highly verbose, redundant, incoherent, and poorly structured. Our current solution approach tackles this challenge on multiple levels. In particular, it (i) resolves syntactic heterogeneity (e.g., varying time stamp formats in different logs) by lifting raw log data and describing it with a rich semantic log vocabulary; (ii) performs entity resolution to uniquely identify entities even in heterogeneous log sources with varying identifiers (e.g., IP addresses vs host names, different user names in different applications, etc.); (iii) tackles semantic heterogeneity (e.g., varying severity scales of different log standards) and offer mappings for an aligned representation; (iv) enables generalization (e.g., Apache and IIS are both web servers).

Figure 3 demonstrates an excerpt of the collected log data in original raw format and Listing 1 shows the JSON-LD output produced by the implemented *log extractor*.

Listing 1: Example log message transformed into JSON-LD

```
{
    "@context":"http://sepses.ifs.tuwien.ac.at/contexts/syslog.jsonld",
    "logMessage":"Apr  9 09:37:47 kabul-VirtualBox systemd[1]: Mounted Huge Pages File System.",
    "timestamp":"2018-04-09T07:37:47.000Z",
    "hasProcessId":"1",
    "hasSeverity":{
        "severityName":"notice",
        "severityCode":"5"
    },
    "@type":"http://purl.org/sepses/vocab/log/sysLog#SysLogEntry",
    "hasLogType":"http://example.org/system#syslog",
    "@id":"http://example.org/logEntry#logEntry-ca1c3894-114f-432d-befd-abca46258e85",
    "hasProgram":{
        "programName":"systemd"
    },
    "logFilePath":"/var/log/syslog",
    "hasFacility":{
        "facilityCode":"1",
        "facilityName":"user-level"
    },
    "input":{
        "type":"log"
    },
    "originatesFrom":{
        "hostName":"kabul-VirtualBox"
    }
}
```

The SPARQL query in Listing 2 demonstrates entity resolution by querying for log events from any host within the defined timeframe.

Listing 2: Query 1

```
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix scl: <http://purl.org/sepses/vocab/log/coreLog#>
prefix sys_bg: <http://purl.org/sepses/bg/system#>

SELECT  ?time ?logType ?hostName ?ipAddress ?hostType ?message
WHERE { ?logEntry a scl:LogEntry;
            scl:originatesFrom ?host;
            scl:hasLogType ?logType;
            scl:logMessage ?message;
            scl:timestamp ?time.
        ?host sys_bg:hostType ?hostType;
            scl:hostName ?hostName;
            scl:ipAddress ?ipAddress.

 FILTER (?time > "2018-04-09T07:29:00+00:00"^^xsd:dateTime  &&
 ?time <"2018-04-09T07:34:00+00:00"^^xsd:dateTime)}
```

Table 1: Result Query 1

time (xsd:dateTime)	logType	hostName	ipAddress	hostType	message
2018-04-09T07:29:15+00:00	scl:syslog	kabul-VirtualBox	192.168.0.164	DatabaseServer	"org.debian.apt[683]: ..."
2018-04-09T07:31:45+00:00	scl:apache	linux-Machine	192.145.0.124	WebServer	*"GET /presentations/"*
2018-04-09T07:31:45+00:00	scl:apache	linux-Machine	192.145.0.124	WebServer	*"GET /presentations/..."*
2018-04-09T07:31:45+00:00	scl:syslog	kabul-VirtualBox	192.168.0.164	DatabaseServer	*"systemd-tmpfiles[3572]: ..."*
2018-04-09T07:31:45+00:00	scl:syslog	kabul-VirtualBox	192.168.0.164	DatabaseServer	*"systemd[1]: Started ..."*

As the result excerpt in Table 1 shows, all events have a host name and IP address. This information comes from the background knowledge, by linking the host node from the log stream (`owl:sameAs`) to the host entity defined in the background knowledge as shown in Figure 4. Silk [33], an open source framework for integrating heterogeneous data sources, is an option to reach this goal. Listing 3 shows an `owl:sameAs` rule (`LinkageRule`) with a `levenshteinDistance` comparison to introduce a link if the host names match.

Listing 3: Silk `owl:sameAs` LinkageRule

```
<LinkageRule linkType="owl:sameAs">
     <Compare id="levenshteinDistance1" required="false" weight="1" metric="levenshteinDistance"
         threshold="0.0" indexing="true">
          <TransformInput id="lowerCase1" function="lowerCase">
            <Input id="sourcePath1" path="/syslog:hostName"/>
          </TransformInput>
          <TransformInput id="lowerCase2" function="lowerCase">
            <Input id="targetPath1" path="/bgk:hostName"/>
          </TransformInput>
          <Param name="minChar" value="0"/><Param name="maxChar" value="z"/>
       </Compare>
</LinkageRule>
```

The host type is also defined in the background knowledge and could be automatically determined based on the software running on the host.

Integration. In the course of monitoring events to identify activities relevant from a security perspective, it is often necessary to correlate isolated indicators from various log sources, hence, an integrated view is required. Our solution combines different log sources into a single log stream with a uniform core and using appropriate vocabularies for

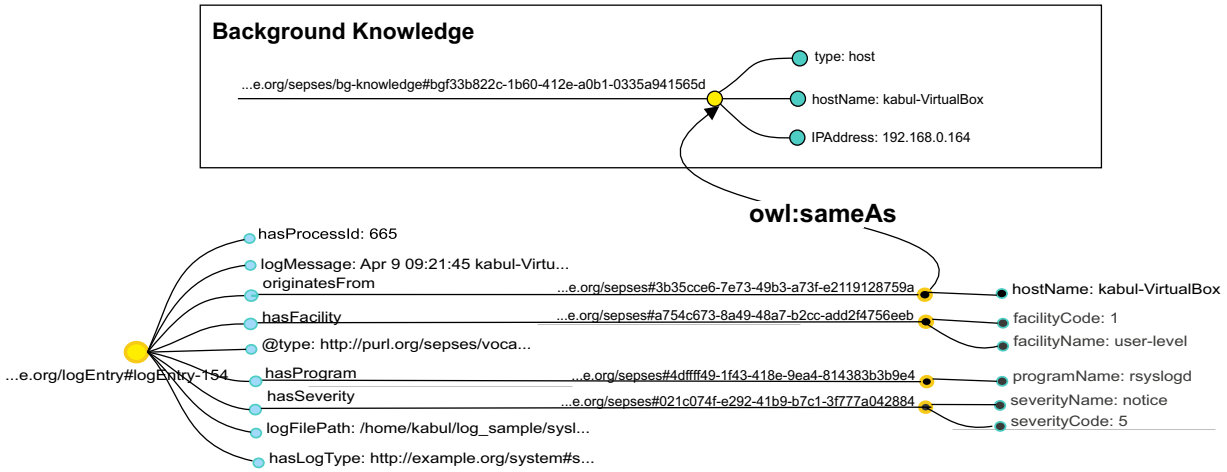

Fig. 4: sameAs linking between an auxiliary node from the log stream and a host entity from background knowledge

Listing 4: Query 2

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix scl: <http://purl.org/sepses/vocab/coreLog#>
prefix syslog: <http://purl.org/sepses/vocab/log/sysLog#>
prefix cve: <http://purl.org/sepses/vocab/cve#>

SELECT ?programName ?CVE ?CVE_Desc
WHERE { ?logEntry a scl:LogEntry;
            scl:originatesFrom ?host;
            syslog:hasProgram ?program.
            ?program cve:hasVulnerability ?CVE;
                     syslog:programName ?programName.
            ?CVE cve:cveDesc ?CVE_Desc.
            ?host scl:hostName "kabul-VirtualBox"^^rdfs:Literal}
```

each log source. Whereas standard SIEM systems also provide some level of integrated access, our approach makes log data machine-readable and semantically integrate-able from the beginning rather than collecting raw log data and integrating it at query time. The query results in Table 1 already illustrate this by combining two disparate log sources (*syslog* and *apache*) and providing integrated results.

Context. Security-related events are typically highly context-specific and hence, their interpretation requires extensive background knowledge. In our approach, we dynamically link rich background knowledge to the extracted log events and their associated entities, respectively.

Listing 4 demonstrates how vulnerability information from Common Vulnerabilities and Exposures (CVE) [20] can be linked to affected servers. Based on the background knowledge on installed software (organization-specific system knowledge) and CVE definitions mapped to software versions, an analyst could ask for all vulnerabilities on a specific host.[7] Naturally, any other background knowledge can be linked and queried in a similar fashion.

Automation. Manual inspection of each log event is generally infeasible due to the vast amount of log data. Therefore, security analysts typically rely on reactive measures, such as alarms triggered by simple rules and subsequent analysis

[7] The integration of formally modeled vulnerability information will be an interesting area for future work

of potential causes. Our approach aims to reduce the burden on analysts by harmonizing various log sources and facilitating highly expressive semantic queries. Based on the log vocabularies introduced in this paper, we plan to develop methods for automated log interpretation and causal linking of events in the future.

4. Related Work

Despite various initiatives to define common logging standards, a variety of log sources, inconsistent log timestamps and content, and inconsistent log formats [12] remain a challenge to this day. In the following, we partition related work into industrial practice, log vocabularies, and approaches focused on semantic log analysis.

Industry. A variety of logging systems are in production use today, such as operating system logs (e.g., syslogd [6] and Windows Event Logs [17]), web server logs (e.g., Extended Log File Format [9], NGINX logging [22], W3C Extended log file format [2]), database logs, firewall logs, etc. To collect and manage log messages from multiple sources, a variety of services have emerged, including Splunk [29], Papertrail [24], Librato [14], and Logstash [32], but they do not strive for semantic integration. Key-value based log formats, such as LOGFMT [28] have become a common approach to format log data. To extract information from raw textual log data, regular expressions are a widely used option.

Log vocabularies. An early initiative to standardize heterogeneous vocabularies to express electronic systems' events in a uniform, device-independent manner is Common Event Expression (CEE), which was driven by MITRE [19]. CEE distinguishes the taxonomy (semantic event type), the log syntax (instance data), and the log transport component for exchange. In 2014, however, MITRE stopped all work on CEE due to discontinued funding of the U.S. Government.

Heterogeneous log file formats have been identified as a serious impediment to the effectiveness of security controls and intrusion detection systems [26]. In the same paper, the authors discuss disadvantages of existing common log formats for normalization of security events and propose a new, extensible log format that is specifically designed for intrusion detection purposes.

In another work that aims to apply semantics to log information [21], the authors focus on web application firewalls and introduce ontologies to avoid ambiguities and vagueness. They take a first step and propose *OntoSeg*, an ontology to model web application firewall logs. The ontology was built in Protégé by inspecting actual log files and identifying major classes and their relationships manually. Compared to our approach, their scope is rather narrow and focused on ontology engineering rather than large-scale RDF processing.

Another ontology-driven approach based on the well-known java logging utility log4j [1] is RLOG [10]. It covers the basic classes for application logging, such as *Log Entry*, *Log Level*, *Status Code* and the fundamental properties, like *Logging message* and *Logging date*. In summary, this ontology comprises some basic concepts, but provides far less than the expressiveness we require. Another ontology, with more depth is the OpenLink Logging Ontology [23]. It uses FOAF, Dublin Core and OWL 2. The SPECIAL Policy Log Vocabulary (Splog) [3] offers a vocabulary for log data that models processing and sharing events that should comply with a given consent provided by a data subject. Due to its specific purpose for consent checking, it does not cover the necessary elements to model common log sources. The log provenance concepts from this vocabulary could be a useful future extension for our approach.

Semantic log analysis. An early approach towards semantic modeling of the logging domain was developed in the context of computer forensics [4, 27]. The authors argue that context information, such as environmental data and configuration information, is vital in forensic analyses. Their overall goal is to develop a general solution (*FORE* - "Forensics of Rich Events") to facilitate human understanding and automated inference. To this end, they define an OWL ontology that captures event information and allows for generalization and composition (e.g., *DownloadExecutionEvent*, which is composed of a *FileReceiveEvent* and an *ExecutionEvent*). The authors develop a custom rule language, based on F-Logic, to express correlation rules. An event browser provides the interface to investigate events and explore detected causalities.

Although FORE aims for a similar goal as our approach (i.e., semantic log analysis), a key difference is its strong focus on forensics. This is reflected in more heavyweight ontology engineering approaches and their choice of technologies. Our approach is grounded in more recently developed Linked Data and Semantic Web technologies such as

JSON-LD and state-of-the-art technologies for scalable log harvesting. This will allow us to implement fast extraction pipelines and potentially allow processing of semantic log streams in (near) real-time, based on recent advances in RDF stream processing technologies [5]. Finally, whereas FORE's ad-hoc ontologies were designed to illustrate particular use cases, we aim to develop a modular vocabulary stack that can cover a variety of different log sources.

Another early approach that tackles the problem that manual review of syslog messages is tedious and error prone is discussed in [30]. The authors apply machine-learning algorithms to detect anomalies in the syslog message stream through classification and investigate cause-effect hypotheses in large multiple-source event log sets. Automatically generated word-granular regular expressions for system event logs are used. They consider their approach as an alternative to expert systems, which require a set of rules that are time-intensive to create and maintain.

A set of publications focuses on log files in connection with web usage mining. Properties of web log data, such as their format, location, and access rules are discussed in [8]. Another Log Ontology (LO) for web usage mining is introduced in [18]. In this ontology, *ComplexEvents* are composed out of *AtomEvents*, which is similar to the idea of extracting high-level events from low-level events discussed in this paper.

5. Conclusions

The SEPSES architecture introduced in this paper aims to create a versatile platform for semantic security log analysis to facilitate effective and efficient security monitoring, auditing, forensics, and incidence response applications.

This platform will support security analysts and complement their human intuition and expertise with machine interpretation of security-related information. Thereby, we aim to reduce the manual work necessary to connect individual pieces of information in disparate log sources and support security analyses by contextualizing, integrating, and linking disparate log information. This will create a semantically explicit, context-rich environment that has the potential to complement existing intrusion detection techniques and increase the precision of alerts and reduce security analysts' "alert fatigue" due to a large number of false positives they tend to generate.

In this paper, we focused on the foundation of the proposed approach to bring meaning to vast volumes of raw textual log data, i.e., log vocabularies, log acquisition, and initial extraction. We outlined how we combine state-of-the-art semantic web and log acquisition technologies and illustrated how the vocabularies provide a foundation for alignment, integration, and linking to background knowledge. Based on a prototypical implementation and illustrative examples, we also highlighted how the proposed approach can tackle current challenges in security analysis.

In future work, we will focus on the interpretation of individual events as well as semantic techniques to establish relations between events and link them to security background knowledge (e.g., vulnerabilities and threat intelligence). Next, we will investigate scalable approaches for the storage and semantic querying of large log archives. To this end, we will evaluate big data approaches and scalable linked data querying interfaces and explore architectural options for scalable semantic log processing infrastructures. Finally, another promising venue for future work on semantic security monitoring is RDF stream processing.

Acknowledgements. This work was sponsored by the Austrian Science Fund (FWF) and netidee SCIENCE under grant P30437-N31, and the COMET K1 program by the Austrian Research Promotion Agency. The authors thank the funders for their generous support.

References

[1] Apache Software Foundation, Accessed: 2018-07-26a. Apache log4j 2. https://logging.apache.org/log4j/2.x/.
[2] Apache Software Foundation, Accessed: 2018-07-26b. Log files. https://httpd.apache.org/docs/1.3/logs.html#common.
[3] Bonatti, P., Dullaert, W., Fernndez, J.D., Kirrane, S., Milosevic, U., Polleres, A., Accessed: 2018-07-26. The special policy log vocabulary. https://aic.ai.wu.ac.at/qadlod/policyLog/.
[4] Clark, A., Mohay, G., Schatz, B., 2004. Rich event representation for computer forensics, in: Kozan, E. (Ed.), Proceedings of the Fifth Asia-Pacific Industrial Engineering and Management Systems Conference (APIEMS 2004), Queensland University of Technology Publications, Gold Coast, Queensland. pp. 2.12.1–2.12.16.
[5] DellAglio, D., Della Valle, E., van Harmelen, F., Bernstein, A., 2017. Stream reasoning: A survey and outlook. Data Science , 1–25.
[6] die.net, Accessed: 2018-07-26. syslogd. https://linux.die.net/man/8/syslogd.
[7] Ghent University, Accessed: 2018-07-26. Enabling linked open data to accomplish the envisaged semantic web. http://rml.io/.

[8] Grace, L.K.J., Maheswari, V., Nagamalai, D., 2011. Web log data analysis and mining, in: Meghanathan, N., Kaushik, B.K., Nagamalai, D. (Eds.), Advanced Computing, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 459–469.
[9] Hallam-Baker, P.M., Behlendorf, B., Accessed: 2018-07-26. Extended log file format. https://www.w3.org/TR/WD-logfile-960221.html.
[10] Hellmann, S., Accessed: 2018-07-26. Rlog - an rdf logging ontology. http://persistence.uni-leipzig.org/nlp2rdf/ontologies/rlog/rlog.html.
[11] IBM Corporation, Ponemon Institute, 2015. 2015 Cost of Data Breach Study: Global Analysis. Ponemon Institute Research Report.
[12] Kent, K., Souppaya, M.P., 2006. SP 800-92. Guide to Computer Security Log Management. Technical Report. Gaithersburg, MD, United States.
[13] Lanthaler, M., 2013. Creating 3rd generation web apis with hydra, in: Proceedings of the 22nd International Conference on World Wide Web, ACM. pp. 35–38.
[14] Librato Inc., Accessed: 2018-07-26. librato. https://www.librato.com/.
[15] Mandiant, 2018. M-Trends 2018. Special report. URL: https://www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf.
[16] McCandless, D., Evans, T., 2018. World's biggest data breaches. http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/ <accessed May 6, 2018>.
[17] Microsoft, Accessed: 2018-07-26. Windows event log. https://msdn.microsoft.com/en-us/library/windows/desktop/aa385780(v=vs.85).aspx.
[18] Ming, S., Shang, Q., Bo, C., 2012. A taxonomic relationship learning approach for log ontology content event. JDIM 10, 109–113.
[19] MITRE, Accessed: 2018-07-26a. Common event expression (cee). https://cee.mitre.org.
[20] MITRE, Accessed: 2018-07-26b. Common vulnerabilities and exposures (cve). http://cve.mitre.org/about/.
[21] do Nascimento, C.H., Ferraz, F.S., Assad, R.E., e Silva, D.L., da Rocha, V.H., 2011. Ontolog: Using web semantic and ontology for security log analysis, in: Proceedings of the Sixth International Conference on Software Engineering Advances (ICSEA 2011), pp. 177–182.
[22] NGINX Inc., Accessed: 2018-07-26. Configuring logging. https://docs.nginx.com/nginx/admin-guide/monitoring/logging/.
[23] OpenLink Software, Accessed: 2018-07-26. Openlink logging ontology. http://www.openlinksw.com/ontology/logging.
[24] Papertrail Inc., Accessed: 2018-07-26. Solarwinds papertrail. https://papertrailapp.com/.
[25] Ponemon Institute, 2014. Exposing the Cybersecurity Cracks: A Global Perspective. Technical Report.
[26] Sapegin, A., Jaeger, D., Azodi, A., Gawron, M., Cheng, F., Meinel, C., 2013. Hierarchical object log format for normalisation of security events, in: 2013 9th International Conference on Information Assurance and Security (IAS), pp. 25–30.
[27] Schatz, B., Mohay, G.M., Clark, A., 2004. Generalising event forensics across multiple domains, in: Valli, C. (Ed.), 2nd Australian Computer Networks Information and Forensics Conference, School of Computer Networks Information and Forensics Conference, Edith Cowan University, Perth, Australia. pp. 136–144. For more information, please refer to the conference website (see hypertext link) or contact the authors.
[28] Setter, M., Accessed: 2018-07-26. Logfmt: A log format thats easy to read and write. https://blog.codeship.com/logfmt-a-log-format-thats-easy-to-read-and-write/.
[29] Splunk Inc., Accessed: 2018-07-26. Splunk. https://www.splunk.com/.
[30] Stearley, J., 2004. Towards informatic analysis of syslogs, in: 2004 IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935), pp. 309–318.
[31] Symantec, 2018. 2018 Internet Security Threat Report. ISTR Volume 23. URL: https://www.symantec.com/security-center/threat-report.
[32] Turnbull, J., 2013. The Logstash Book. James Turnbull. Available at https://logstashbook.com.
[33] Volz, J., Bizer, C., Gaedke, M., Kobilarov, G., 2009. Silk A Link Discovery Framework for the Web of Data. Technical Report. Madrid, Spain.