Automated Knowledge Graph Construction From Raw Log Data*

Andreas Ekelhart[1][0000−0003−3682−1364], Fajar J. Ekaputra[2][0000−0003−4569−2496], and Elmar Kiesling[1][0000−0002−7856−2113]

[1] WU (Vienna University of Economics and Business), Welthandelsplatz 1, 1020 Vienna, Austria first.last@ai.wu.ac.at

[2] TU Wien (Vienna University of Technology), Favoritenstraße 9-11/194, 1040 Vienna, Austria fajar.ekaputra@tuwien.ac.at

Abstract. Logs are a crucial source of information to diagnose the health and status of systems, but their manual investigation typically does not scale well and often leads to a lack of awareness and incomplete transparency about issues. To tackle this challenge, we introduce SLOGERT, a flexible framework and workflow for automated construction of knowledge graphs from arbitrary raw log messages. To this end, we combine a variety of techniques to facilitate a knowledge-based approach to log analysis.

1 Introduction

Log files are a vital source of run time information about a system's state and activities in many areas of information systems development and operations, e.g., in the context of security monitoring, compliance auditing, forensics, and error diagnosis.

Around those varied applications, a market for log management solutions has developed that assist in the process of storing, indexing, and searching log data – the latter typically through some combination of manual inspection and regular expressions to locate specific messages or patterns [3]. Commercially available log management solutions (e.g., Splunk[3] or Logstash[4]) facilitate aggregation, normalization, and storage, but provide limited integration, contextualization, linking, enrichment, and querying capabilities. Consequently, although they ease manual analytical processes somewhat, investigations across multiple heterogeneous log sources with unknown content and message structures remains a challenging and time-consuming task. Analysts therefore typically have to cope with

* This work was sponsored by the Austrian Science Fund (FWF) and netidee SCIENCE under grant P30437-N31, and the Austrian Research Promotion Agency FFG under grant 877389 (OBARIS). The authors thank the funders for their generous support.

[3] https://splunk.com

[4] https://logstash.net

many different types of events, expressed with different terminologies, and represented in a multitude of formats [2], particularly in large-scale systems composed of heterogeneous components.

In this paper, we propose SLOGERT (*Semantic LOG ExtRaction Templating*), a workflow for automated knowledge graph construction from unstructured, heterogeneous, and potentially fragmented log sources. SLOGERT combines extraction techniques that leverage particular characteristics of log data into a modular and extensible processing framework. In particular, we propose a workflow that combines log parsing and event template learning, natural language annotation, keyword extraction, automatic generation of RDF graph modelling patterns, and linking and enrichment to extract and integrate the evidence-based knowledge contained in logs. By making log data amenable to semantic analysis, the workflow fills an important gap and opens up a wealth of data sources for knowledge graph building.

2 Building knowledge graphs from log files

In this section, we introduce the SLOGERT[5] architecture, components, and implementation. The resulting workflow, illustrated in Figure 1, expects unstructured log files as input and consists of five phases:

1. *Template and Parameter Extraction* Log files typically consist of structured elements (e.g., time stamp, device id, facility, message severity), and an unstructured free-text message. To extract log templates from such raw unstructured logs, we use LogPAI [5] to identify constant strings and variable values in the free-text message content. This results in two files, i.e., *(i)* a list of log templates discovered in the log file, marking the position of variable parts (parameters), and *(ii)* the content of the logs, with each log line linked to one of the log template ids, and the extracted instance parameters as an ordered list. At the end of this stage, we have templates and extracted (variable) parameters, but their semantic meaning is yet undefined.
2. *Semantic Annotation* receives the log templates and the instances with the extracted parameters as input. This phase consists of two sub-phases:
   (a) *Semantic template annotation* initiates the parameter type detection by first selecting a set of log lines for each template and then applying rule-based Named Entity Recognition (NER) techniques. Specifically, we use the TokensRegex Framework from Stanford CoreNLP[6] to define sequences of tokens, and map them to semantic objects. As log messages often do not follow the grammatical rules of natural language expressions (e.g., URLs, identifiers), we additionally apply standard Regex patterns on the complete message. For each detection pattern, we define a type and a property from a log vocabulary to use for the detected entities. To generate a consistent representation over heterogeneous log files, we

[5] Project website `https://w3id.org/sepses/index.php/slogert/`
[6] `https://nlp.stanford.edu/pubs/tokensregex-tr-2014.pdf`

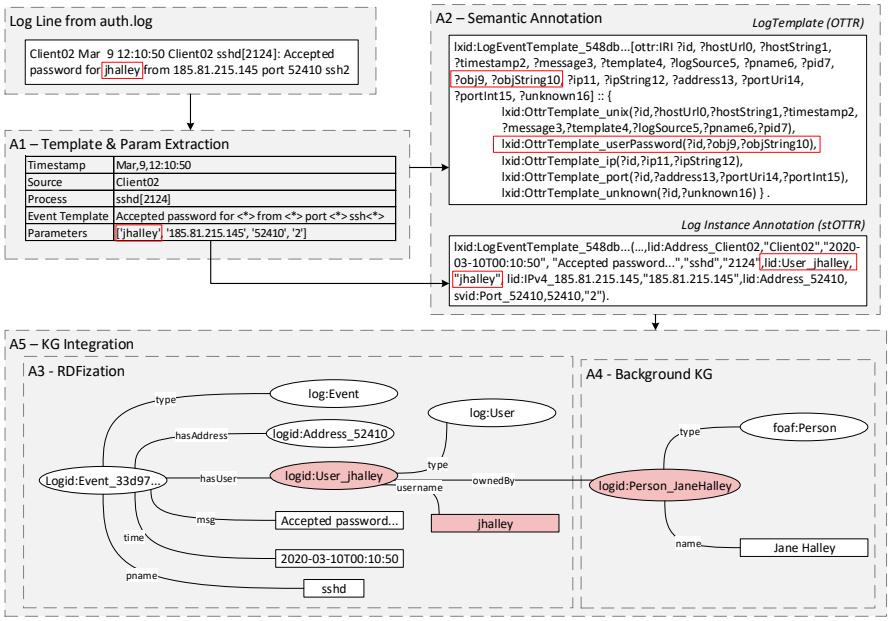

Fig. 1: Example: Processing of a single log line (user entity highlighted)

extended an existing log vocabulary[7] and mapped it to the Common Event Expression (CEE) [2] taxonomy as shown in Figure 2. Once we have identified each parameter of a template, we generate Reasonable Ontology Templates (OTTRs) [4].

(b) *Semantic instance annotation* receives a set of annotated templates from the semantic template annotation process. Based on these annotated templates, we transform all log line instances into stOTTR, a custom file format similar to Turtle, which references the OTTR templates. In addition, we apply the CoreNLP Annotation features to extract keywords from each log message to provide additional context.

3. *RDFization* generates a knowledge graph for each log file based on the OTTR templates and stOTTR instance files generated in the extraction component. To this end, we integrate Lutra[8], the reference implementation for the OTTR language to expand all instances into regular RDF graphs.
4. *Background Knowledge Graph (KG) linking* contextualizes entities that appear in a log file with background knowledge. We distinguish local background knowledge (e.g., employees, servers, installed software, and documents) and external background knowledge (e.g., publicly available cybersecurity information from [1]).

[7] `https://w3id.org/sepses/vocab/log/core`
[8] `https://ottr.xyz/#Lutra`

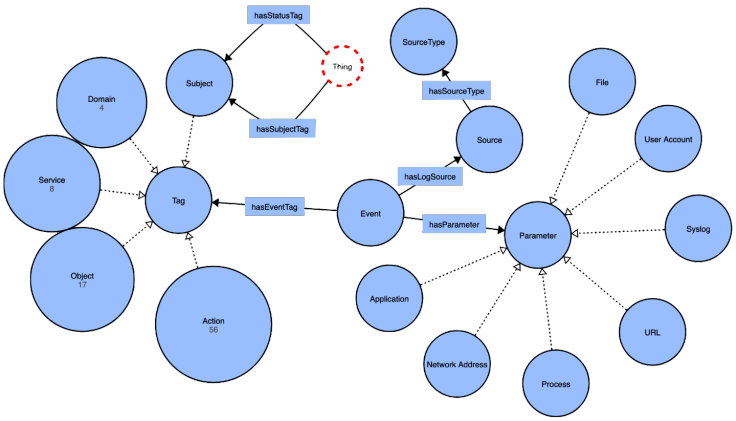

Fig. 2: An excerpt of the extended log ontology

5. *Knowledge Graph Integration* combines the KGs generated from the previously isolated log files and sources into a single, linked representation. In our prototype, the generated KGs and the background knowledge share the same vocabulary and hence, can be easily merged together.

3 Example

To illustrate the proposed approach, we simulated user behavior in a virtual lab network in the Azure platform according to scripted scenarios and collected various log files from each host (e.g., auth, sys, vsftpd). We then generated an integrated log graph by processing the log files with our prototype.

Listing 1 illustrates how to query the integrated graph, which contains events from different sources and hosts, for log events which have a connected IP and port number, and enrich the result by showing the services that are typically running on those ports. This background information comes from a referenced ontology on services and ports[9]. Figure 3 provides an excerpt of the query results.

4 Conclusions

This paper introduced SLOGERT, a flexible framework for automated knowledge graph construction from unstructured log data. In our own research, we will first apply the proposed approach in the context of semantic security analytics, but we see more general potential for the approach to drive adoption of Semantic Web technologies in domains where log data needs to be analyzed.

The results of the current prototype are promising – we next plan to study the quality of the automatically generated log graphs. This includes evaluating

[9] `https://w3id.org/sepses/id/slogert/port-services`

```
PREFIX...# Prefixes omitted for the sake of brevity
SELECT ?time ?event ?sourceLabel ?ipLabel ?portNumber ?serviceName
WHERE {
    ?event log:time ?time; log:hasSource ?source ; log:hasPort ?port ; log:hasIP ?ip .
    ?source log:hasSourceType ?sourceType .
    ?sourceType rdfs:label ?sourceLabel .
    ?ip log:ipv4 ?ipLabel .
    ?port log:port ?portNumber ; log:port ?portNumber ; log:linkedPortService ?linkedPort .
    ?portProtocolCombination svid:hasPort ?linkedPort ; svid:hasService ?service .
    ?service rdfs:label ?serviceName . } ORDER BY ASC(?time)
```

Listing 1: SPARQL query to get log events with ports and their standard services

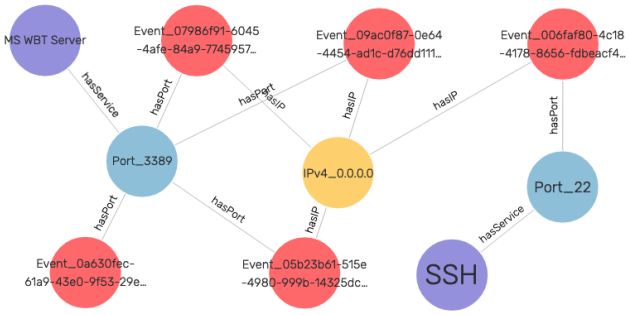

Fig. 3: Events with ports and their standard services (excerpt)

the correct detection of parameters in log lines, as well as the correct entity detection and linking, also on unknown log sources. The completeness and quality of the extracted keywords could also be evaluated in user studies and inform extensions of the applied method. Furthermore, we will focus on graph management for template evolution and incremental updating of log knowledge graphs in future work. Finally, we plan to compare analyst workflows in commercial log management tools with our solution to highlight the advantages of semantic graphs in log analysis, and to identify potential for improvement and synergies.

References

1. Kiesling, E., Ekelhart, A., Kurniawan, K., Ekaputra, F.: The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity. In: The Semantic Web – ISWC 2019, pp. 198–214. Springer (2019)
2. MITRE: About Common Event Expression, `https://cee.mitre.org`
3. Oliner, A., Ganapathi, A., Xu, W.: Advances and challenges in log analysis. Communications of the ACM **55**(2), 55–61 (2012)
4. Skjæveland, M.G., Lupp, D.P., Karlsen, L.H., Forssell, H.: Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In: The Semantic Web – ISWC 2018. pp. 477–494. Springer (2018)
5. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Tools and benchmarks for automated log parsing. In: 41st Int. Conf. on Software Engineering: Software Engineering in Practice. p. 121–130. IEEE Press (2019)