Cross-Platform File System Activity Monitoring and Forensics – A Semantic Approach

Kabul Kurniawan, Andreas Ekelhart, Fajar Ekaputra, Elmar Kiesling

▶ **To cite this version:**

Kabul Kurniawan, Andreas Ekelhart, Fajar Ekaputra, Elmar Kiesling. Cross-Platform File System Activity Monitoring and Forensics – A Semantic Approach. 35th IFIP International Conference on ICT Systems Security and Privacy Protection (SEC), Sep 2020, Maribor, Slovenia. pp.384-397, 10.1007/978-3-030-58201-2_26 . hal-03440824

HAL Id: hal-03440824

https://inria.hal.science/hal-03440824v1

Submitted on 22 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Cross-Platform File System Activity Monitoring and Forensics – A Semantic Approach

Kabul Kurniawan*[1,3][0000−0002−5353−7376], Andreas Ekelhart[1,2][0000−0003−3682−1364], Fajar Ekaputra[1][0000−0003−4569−2496], and Elmar Kiesling[1][0000−0002−7856−2113]

[1] TU Wien, Favoritenstraße 9-11, Vienna, Austria
[2] SBA Research, Floragasse 7, Vienna, Austria
[3] University of Vienna, Währingerstraße 29, Vienna, Austria

Abstract. Ensuring data confidentiality and integrity are key concerns for information security professionals, who typically have to obtain and integrate information from multiple sources to detect unauthorized data modifications and transmissions. The instrumentation that operating systems provide for the monitoring of file system level activity can yield important clues on possible data tampering and exfiltration activity but the raw data that these tools provide is difficult to interpret, contextualize and query. In this paper, we propose and implement an architecture for file system activity log acquisition, extraction, linking and storage that leverages semantic techniques to tackle limitations of existing monitoring approaches in terms of integration, contextualization, and cross-platform interoperability. We illustrate the applicability of the proposed approach in both forensic and monitoring scenarios and conduct a performance evaluation in a virtual setting.

Keywords: Semantic log analysis · Digital forensics · File system monitoring · Exfiltration detection

1 Introduction

In our increasingly digitized world, Information and Communication Technologies pervade all areas of modern life. Consequently, organizations face difficult challenges in protecting the confidentiality and integrity of the data they control, and theft of corporate information – i.e., data breaches or data leakage – have become a critical concern [7].

In the face of increasingly comprehensive collection of sensitive data, such incidents can become an existential threat that severely impacts the affected organization, e.g., in terms of reputation loss, decreased trustworthiness, and direct consequence that affect their bottom line. Fines and legal fees, either due to contractual obligations or laws and regulations (e.g., the General Data

* {first.last}@tuwien.ac.at

Protection Regulation in the EU), have become another critical risk. Overall, the number and size of data breaches have been on the rise in recent years years[4].

On a technical level, exfiltration of sensitive data is often difficult to detect. In this context, we distinguish two main types of adversaries and associated threat models: *(i)* an insider with legitimate access to data, who either purposely or accidentally exfiltrates data, and *(ii)* an external attacker who obtains access illegitimately. Insiders typically have multiple channels for exfiltration at their disposal, including conventional protocols (e.g., ftp, sftp, ssh, scp), cloud storage services (e.g., dropbox, onedrive, google drive, WeTransfer), physical media (e.g., USB, laptop, mobile phone), messaging and email applications, and dns tunneling [11]. Whereas an insider may leverage legitimate access permissions directly or at least internal resources as a starting point, an external attacker must first infiltrate the organization network and obtain access to the data (e.g., by spreading malware or spyware, stealing credentials, eavesdropping, brute forcing employee passwords, etc.).

State-of-the-art perimeter security solutions such as intrusion detection and prevention systems (IDS/IPS), firewalls, and network traffic anomaly detection are per se generally not capable of detecting insider attacks [20]. However, such activities typically leave traces in the network and on the involved systems, which can be used to spot potential misuse in real time or to reconstruct and document the sequence of events associated with an exfiltration and its scope ex-post. This examination, interpretation, and reconstruction of trace evidence in the computing environment is part of digital forensics. Upon detection of security violations, forensic analysts attempt to investigate the relevant causes and effects, frequently following the hypothesis-based approach to digital forensics [6]. Although there are a variety of tools and techniques available that are employed during a digital investigation, the lack of integration and interoperability between them, as well as the formats of their sources and resulting data hinder the analysis process [8].

In this paper, we introduce a novel approach that leverages semantic web technologies to address these challenges in the context of file system activity analysis. This approach can harmonize heterogeneous file and process information across operating systems and log sources. Furthermore, it provides contextualization through interlinking with relevant information and background knowledge.

The research question we address in this article is: *How can semantic technologies support digital file activity investigations?* Addressing this question resulted in the following main contributions: *(i)* a set of log and file event vocabularies (Section 3); *(ii)* an architecture and prototypical implementation for file system log acquisition, event extraction, and interlinking across heterogeneous systems and with background knowledge (Section 4); *(iii)* a set of demonstration scenarios for continuous monitoring and forensic investigations (Section 5); and *(iv)* a performance evaluation in a virtual setting (Section 6).

[4] https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

2 Related Work

Our approach builds upon and integrates multiple strands of work, which we will review in the following: *(i)* approaches for file activity monitoring, both in the academic literature and commercial tools; *(ii)* file system ontologies; and *(iii)* semantic file monitoring & forensics.

File Activity Monitoring In contrast to the approach presented in this paper, prior work in this category does not involve semantic or graph-based modeling, which facilitates interoperability and integration, contextualization through interlinking with background knowledge, and reasoning.

The authors in [12] focus on data exfiltration by insiders. They first apply statistical analyses to characterize legitimate file access patterns and compare those to file access patterns of recent activities to identify anomalies. The authors mention that the approach can result in a high number of suspicious activities, which can be impractical for individual investigation. [4] aims to predict insider threats by monitoring various parameters such as file access activity, USB storage activity, application usage, and sessions. In their evaluation, they train a deep learning model on legitimate user activity and then use the model to assign threat scores to unseen activities. In [3], the authors introduce a policy-based system for data leakage detection that utilizes operating system call provenance. They facilitate real-time detection of data leakage by tracking operations performed on sensitive files. This approach is similar to the one presented in this paper in its objectives, i.e., it also aims to monitor file activities (copy, rename, move), but it does not cover contextualization and linking to background knowledge. [9] proposes an approach that leverages data provenance information from OS kernel messages to detect exfiltration of data returned to users from a database. The proposed system builds profiles of users' actions to determine whether actions are consistent with the tasks of the users. While it has similar goals, the focus is limited on data exfiltration from databases via files.

Apart from the academic research on various techniques for file activity monitoring, a wide range of tools is available commercially, such as Solarwind Server and Application Monitor, ManageEngine DataSecurity Plus, PA File Insight, STEALTHbits File Activity Monitor, and Decision File Audit. These tools cover varying scopes of leakage detection and typically provide a simple alerting mechanism upon suspicious activity. Another category of existing tools are Security Information and Event Management systems (e.g., LogDNA, Splunk, ElasticSearch). Their purpose is to manage and analyze logs and they do not specifically tackle the problem of tracking file activity life-cycles.

File System Ontologies Ontological representation of file system information has been explored, e.g., in [18], in which the authors propose TripFS, a lightweight framework that applies Linked Data principles for file systems in order to expose their content via dereferenceable HTTP URIs. The authors model file systems with their published vocabulary that is aligned with the NEPOMUK

File Ontology (NFO)[5]. Similar to TripFS, [19] proposes VDB-FilePub to expose file systems as Linked Data and to publish user-defined content metadata. With focus on end-user access, [17] provide an extension to TripFS which enables users to navigate the published files, and to annotate and download them via common web browsers without the need to install special software packages.

In recent work, the authors of [16] proposed a Semantic File System (SFS) Ontology[6] which extends terms from the NEPOMUK ontology. They further provide technical definitions of terms and a class hierarchy with persistent URIs and content negotiation capabilities. In our approach, we use the basic concepts for files, such as file names and file properties as proposed in the related work, but our approach integrates additional concepts, such as, e.g., file activities, source and target locations, and file classification.

Semantic Approaches to File Access Monitoring & Forensics The application of semantics for digital forensics has been the topic of multiple research publications. While they are motivated by similar challenges, such as heterogeneity, variety and volume of data, they do not focus on file activity monitoring and life-cycle construction in particular, but on the digital evidence process in general.

Early work on using semantic web technology in the context of forensics includes [13], which introduces an evidence management methodology to semantically encode why evidence is considered important. An ontology is used to describe the metadata file contents and events in a uniform and application-independent manner. In [1], the authors propose a similar ontology-based framework to assist investigators in analyzing digital evidence. They motivate the use of semantic technologies in general and discuss the advantage of ontological linking, annotations, and entity extraction. A broader architecture to lift the phases of a digital forensic investigations to a knowledge-driven setting is proposed in [8]. This results in an integrated platform for forensic investigation that deals with a variety of unstructured information (e.g., network traffic, firewall logs, and files) and builds a knowledge base that can be consulted to gain insights from previous cases via SPARQL queries.

Finally, in a recent contribution [2], the authors propose a framework that supports forensic investigators during the analysis process. This framework extracts and models individual pieces of evidence, integrates and correlates them using a SWRL rule engine, and persists them in a triplestore. Compared to our approach, their focus is on text processing while file activity analysis is not considered.

The approach presented in this paper extends preliminary work published in [15] by introducing cross-platform interoperability, scenarios that demonstrate the approach, linking to background knowledge and a performance evaluation.

[5] http://oscaf.sourceforge.net/nfo.html
[6] https://w3id.org/sfs-ontology#

3 Conceptualization

Operating systems typically provide mechanisms and instrumentation to obtain information on system-level file system operations, typically on the level of kernel calls. Reconstructing the corresponding user activities, such as editing, moving, copying or deleting a file from these low-level signals can be challenging. In particular, the sequence of micro-operations triggered by a file system operation varies across operating systems and applications, which complicates the analysis. On Windows systems, for instance, file operations such as `Create` generate a number of access operations including `ReadAttributes`, `WriteData`, `ObjectClosed`, etc.

To construct our vocabularies, we analyzed the structure, format, and access patterns of the different file activity log sources on both Windows and Linux. Furthermore, as contextualization is a key requirement for the interpretation of file activity in forensic analyses, we also include sources of *(i)* process activity information, and *(ii)* authentication events (login, logout, etc.) . The scenarios in Section 5 illustrate how we make use of process information and authentication information. Due to space restrictions, we will not cover the process and authentication vocabulary in full detail and refer the interested reader to the source[7].

3.1 Vocabulary

As existing ontologies (reviewed in Section 2) do not fully cover the requirements of our approach, we developed a custom ontology. We followed a bottom-up approach starting from low-level information from log sources with the goal to choose and collect appropriate terms directly from the sources of evidence (e.g. users, hosts, files). We organize our semantic model into two levels, i.e., *log entry* level and *file operation* level. On the *log entry* level, we define a vocabulary to represent information on micro-level operations for both Windows and Linux OS log sources which is based on a previously developed vocabulary [10] for generic log data. On the *file operation* level, we model a generic vocabulary to express higher-level events such as actual file event activity (e.g., created, modified, copied, rename, delete) derived from micro-level operations.

Log Entry vocabularies The Windows Log Event (wle) vocabulary[8] represents Windows file access events using `wle:WindowsEventLogEntry`, a subclass of `cl:LogEntry` from the SEPSES core log[9]. The `wle:Subject` class represents account information such as `wle:accountName` and `wle:logonID`; the `wle:AccessRequest` class represents file access information such as `wle:accessMask` and `wle:accesses`; the `wle:Process` class represents running processes and the `wle:Object` class represents object file information such as `wle:objectName`,

[7] https://w3id.org/sepses/vocab/event/process-event
[8] https://w3id.org/sepses/vocab/log/win-event
[9] https://w3id.org/sepses/vocab/log/core

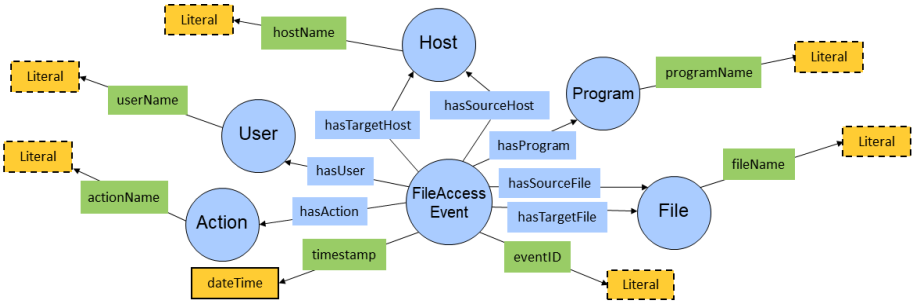

Fig. 1: High-Level event vocabularies (File Access Event)

`wle:objectType`, and `wle:handleID`. To cover Linux file access events, we developed the Linux Log Event (lle)[10] vocabulary that comprises five main classes: `lle:LinuxEventLogEntry`, a subclass of `cl:LogEntry` from the SEPSES core vocabulary, `lle:Event` class, which covers information on file access events such as `lle:eventType`, `lle:eventId`, `lle:eventCategory`, and `lle:eventAction`; the `lle:File` class represents information about file objects such as `lle:fileName` and `lle:filePath`; the `lle:User` class covers information on users who perform the file event activities such as `lle:userName` and `lle:userGroup`; the `lle:Host` class represents `lle:hostArchitecture`, `lle:hostOS`, `lle:hostName`, `lle:hostId`, etc.

The ***File Operation vocabulary***[11] describes `fae:FileAccessEvents` by means of the following properties: `fae:hasAction` reflects the type of access (e.g., created, modified, copied, renamed, deleted); `fae:hasUser` links the file event to the user accessing the file; `fae:hasProgram` represents the executable used to access the file, and `fae:timestamp` captures the time of access. The properties `fae:hasSourceFile` and `fae:hasTargetFile` model the relation between an original and copied instance of a file. Finally, property `fae:hasSourceHost` and `fae:hasTargetHost` represent the hosts where the source and target files are located.

3.2 Background Knowledge

To support contextualization and enrichment, we leverage several existing sources of internal and external background knowledge.

Internal background knowledge can be developed by manually or automatically collecting an organization's persistent information (e.g. IT Assets, Network Infrastructure, Users). In our scenarios, we use predefined internal background knowledge to contextualize and create linking with file access events during event extraction.

[10] https://w3id.org/sepses/vocab/log/linux-event
[11] https://w3id.org/sepses/vocab/event/file-access

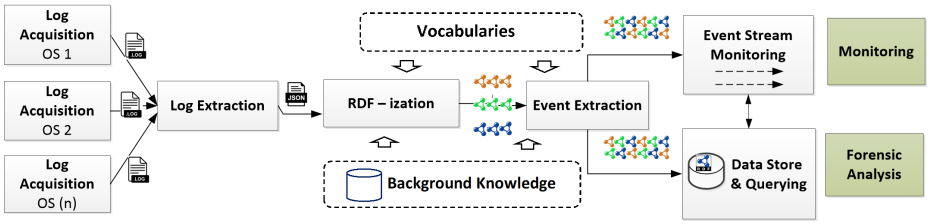

Fig. 2: Solution architecture

Furthermore, it is possible to leverage existing external knowledge, such as the SEPSES cybersecurity knowledge graph (CSKG)[12], to link external information with system events.

4 Architecture & Prototype Implementation

In this section, we describe our architecture and prototypical implementation for semantic integration, monitoring, and analysis of file system activity as depicted in Figure 2.

The ***Log Acquisition*** component deals with the acquisition of log information and is installed as an agent on clients or servers. We implement our Log Acquisition component on *Filebeat*[13], an open-source log data acquisition tool that ships log data from a host for further processing. Using Filebeat, we can easily select and configure and add log sources from both Windows and Linux machines. Furthermore, we use the Filebeat Audit module to ship process and authentication information from the log sources.

The ***Log Extraction*** component handles the parsing of various log data provided by the Log Acquisition component and can act as a filter that keeps only relevant parts. We use *Logstash*[14], an open source log processing tool that provides options for developing processing pipelines to distinguish and handle different types of log sources. Furthermore, it provides different output options such as a web socket protocol that supports data streaming.

The ***RDF-ization*** component transforms data into RDF by mapping structured log data produced by the Log Extraction component to a set of predefined ontologies (cf. Section 3). This produces an RDF graph as the basis of file operation events extraction. We use TripleWave[15] to publish RDF streaming data through specified mappings (e.g. RML[16]). Furthermore, TripleWave supports the web socket protocol to publish the output.

The ***Event Extraction*** component generates file operation events by identifying a sequence of low level (e.g., kernel-level) file system events. Furthermore,

[12] http://sepses.ifs.tuwien.ac.at
[13] https://www.elastic.co/products/beats/filebeat
[14] https://www.elastic.co/products/logstash
[15] https://streamreasoning.github.io/TripleWave/
[16] http://rml.io

it enriches the events by creating links between file operation events and existing internal (hosts, users, etc.) and external (e.g., the SEPSES cybersecurity knowledge graph [14]) background knowledge. We developed a Java-based event extractor[17] and use the C-Sprite[5] engine to implement the event extraction process. C-Sprite is an RDF stream processing engine that allows us to register a set of continuous SPARQL-Construct queries against the low level RDF graph of file system events to generate a graph of file operation events.

Finally, the ***Data Storage, Querying, and Visualization*** component stores the extracted RDF graph of file operation events in a persistent storage (e.g., a triplestore) and facilitates querying and further analysis. We choose the widely-used Virtuoso[18] triple store, which provides a SPARQL endpoint, for our prototypical implementation. Furthermore, we developed a simple web-based graph visualization interface[19] that helps analysts to interpret file access lifecycles (cf. Section 5 for an example).

5 Application Scenarios

In this section, we demonstrate the feasibility of our approach by means of two application scenarios. For both scenarios, we set up a virtual lab with several Windows and Linux machines, users, groups, and shared folders.

5.1 Scenario 1: Data Exfiltration

In the first scenario, we assume that an organization has learned that confidential information was leaked. The task in this scenario is to investigate how and by whom this information has been transferred out of the organizational network.

Figure 3 depicts an excerpt of the company network, including Linux and Windows workstations and a Linux file server that stores company-wide shared data as well as confidential data with restricted access permissions (e.g., customer and financial data). The organization's access model distinguishes two groups: *manager* and *office users*. Both groups are authorized to log in to the company workstations and access the internal file shares. Access to the confidential data is restricted to the *manager* group.

As a starting point, the analyst has the name of a file that contains the leaked sensitive information and starts to investigate its history. Listing 1.2 depicts the SPARQL query to obtain lifecycle information for this file. The result is given in Table 1 and shows that the file *cstcp001.xls* was accessed and modified multiple times. Inspecting the timeline, we can see that a file *customer.xls* was modified on *FileServer1* with the IP *193.168.1.2*. It thereafter was copied, renamed and modified on the file server. Then, the file appeared on *Workstation2* and got deleted from the file server. Finally, the file was renamed to *cstcp001.xls* and copied to another folder on *Workstation2* with the name *Dropbox* in its file path. Figure 4 visualizes the file history.

[17] https://github.com/kabulkurniawan/fileAccessExtractor
[18] https://virtuoso.openlinksw.com/
[19] https://w3id.org/sepses/sparqlplus

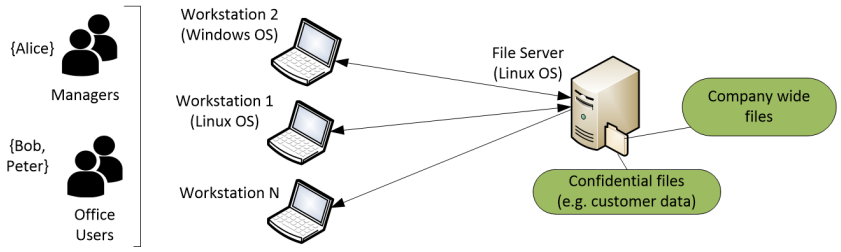

Fig. 3: Scenario 1 network excerpt

```
SELECT distinct ?time ?accessType ?sourceFile ?targetFile ?hostIP ?hostType
    WHERE {
          ?y fae:timestamp ?timestamp.
          ?y fae:hasAction/fae:actionName ?accessType.
          ?y fae:hasSourceFile/fae:pathName ?sourceFile.
          ?y fae:hasTargetFile/fae:pathName ?targetFile.
          ?y fae:hasTargetHost ?h.
          ?h cl:IpAddress ?hostIp.?h fae:hasSourceHost ?hostName.
          ?y fae:hasSourceFile/fae:fileName "cstcp001.xls".
          ?x fae:relatedTo* ?y .
      } ORDER BY ASC(?time)
```

Listing 1.1: SPARQL query to retrieve the history of a file

timestamp	accessType	sourceFile	targetFile	hostIP	hostName
11:06:55	Modified	/home/alc/secdt/customer.xls	/home/alc/secdt/customer.xls	193.168.1.2	FileServer1
13:39:01	Copied	/home/alc/secdt/customer.xls	/home/alc/customer.xls	193.168.1.2	FileServer1
13:39:35	Renamed	/home/alc/customer.xls	/home/alc/customer-cp.xls	193.168.1.2	FileServer1
13:40:23	Modified	/home/alc/customer-cp.xls	/home/alc/customer-cp.xls	193.168.1.2	FileServer1
13:43:17	Created	C:\Work\customer-cp.xls	C:\Work\customer-cp.xls	193.168.2.2	Workstation2
13:43:52	Deleted	/home/alc/customer-cp.xls	/.trash/customer-cp.xls	193.168.1.2	FileServer1
15:50:57	Renamed	C:\Work\customer-cp.xls	C:\Work\cstcp001.xls	193.168.2.2	Workstation2
15:53:52	Copied	C:\Work\cstcp001.xls	C:\DropBox\cstcp001.xls	193.168.2.2	Workstation2

Table 1: File History Results

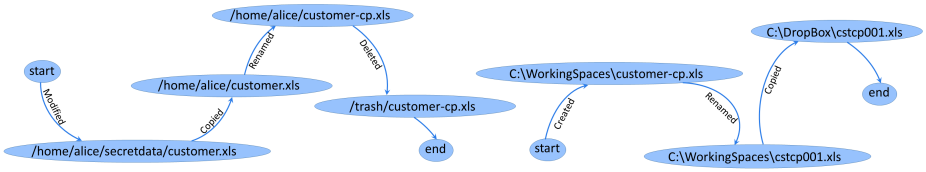

Fig. 4: Graph visualization of the file history

timestamp	eventType	hostIP	hostName	programName	pid	userName	groupName
13:43:17	ProcessStopped	193.168.1.2	FileServer1	/usr/bin/scp	223	Alice	Manager
13:43:17	ProcessStopped	193.168.1.2	FileServer1	/usr/bin/ssh	224	Alice	Manager
13:43:17	ProcessStarted	193.168.2.2	Workstation2	C:\...\sshd.exe	1988	-	-
13:43:18	ProcessStopped	193.168.2.2	Workstation2	C:\...\sshd.exe	1988	-	-

Table 2: Potential Exfiltration Process – Results

Next the analyst wants to know how the file was transferred from *FileServer1* to *Workstation2*. A SPARQL query[20] lists the running processes and user names in the time period of the suspicious activities. Potential exfiltration processes are modeled in the background knowledge with the concept *sys:potentialExfitration-Processes*, which includes channels such as FTP, SCP, SSH, etc. This illustrates how queries can automatically make use of modeled background knowledge. Table 2 shows the results of the query. From this, the analyst learns that a secure copy event */usr/bin/scp* was started on *FileServer1* prior to the file copy and also on the Windows host *Workstation2*. The processes on the file server were performed by user *Alice* from the manager group. The analyst concludes that the *customer-cp.xls* file was successfully transferred via SCP (SSH service) by the user *Alice*.

Next, the analyst wants to collect more information about this file transfer and the users involved in those steps. Therefore, a LoginProcess[21] query is executed to retrieve a list of users logged in to these hosts in the time period of interest, including `userName`, `sourceIp`, `targetIp`, `hostName`, and the `timestamp`. The query result depicted in Table 3 shows that *Alice* was not logged in to *Workstation1* during this time. Instead, *Bob* shows up several times in the login list of Workstation1. From *Workstation1*, a login event was performed on *FileServer1* with Alice's credentials. At the time the file copy to the Dropbox folder happened on *Workstation2*, only Bob was logged in on this computer. Concluding from this evidence, the analysts suspects that Bob logged in to *Workstation1*, then accessed the confidential file on *FileServer1* with the credentials of Alice. Finally, he copied the file to *Workstation2* and exfiltrated the data via Dropbox.

timestamp	eventType	sourceHost	sourceIp	targetHost	targetIp	userName
13:30:23	Login	-	172.24.66.19	Workstation1	192.168.2.1	Bob
13:33:31	Login	-	172.24.66.19	Workstation1	192.168.2.1	Bob
13:38:16	Login	Workstation1	192.168.2.1	FileServer1	192.168.1.2	Alice
14:53:06	Login	-	172.24.66.19	Workstation2	192.168.2.2	Bob

Table 3: Login process results

5.2 Scenario 2: Sensitive data on vulnerable hosts

fileName	hostName	OSName	hostIP	cveId	conf	score
C:\Documents\Customer.xls	Workstation2	Windows	192.168.2.1	2016-1653	COMPLETE	9.3
/home/docs/employee.xls	Workstation3	Linux	192.168.2.1	2016-1583	COMPLETE	7.2

Table 4: Vulnerability assessment results excerpt

In the second scenario, we illustrate how the semantic monitoring approach can be used to protect confidential information by combining public vulnerabil-

[20] https://w3id.org/sepses/IFIP2020/queries/potentialExfitrationProcesses.sparql
[21] https://w3id.org/sepses/IFIP2020/queries/loginProcess.sparql

```
SELECT * WHERE {
    ?s rdf:type fae:FileAccessEvent;
        fae:hasFileAccessType sys:Created;
        fae:hasSourceFile/fae:fileName ?filename;
        asset:hasDataClassification sys:Private;
        fae:hasSourceHost/fae:hostName ?hostName;
        {SELECT ?hostName ?OSName ?hostIP ?cveId ?conf ?score WHERE {
                ?t rdf:type sys:Host. ?t sys:hostName ?hostName.
                ?t sys:OSName ?OSName. ?t sys:IPAddress ?hostIP.
                ?t sys:hasProduct ?p.
            SERVICE <http://sepses.ifs.tuwien.ac.at/sparql> {
                ?cve cve:hasCPE ?p. ?cve cve:id ?cveId.
                ?cve cve:hasCVSS2BaseMetric ?cvss2. ?cvss2 cvss:confidentialityImpact ?conf.
                ?cvss2 cvss:baseScore ?cvssScore. }}}}
```

Listing 1.2: Query to check vulnerable host

ity information with file activity information from inside the company network. We assume a policy that restricts handling of confidential files on hosts with known vulnerabilities. The objective in this scenario is to automatically detect violations of this policy. More precisely, the goal is to spot whenever files flagged as *confidential*[22] are copied or created on an internal host with a known vulnerability.

As background knowledge, we import information on installed software on each host. This information is represented in the Common Platform Enumeration (CPE) format and can be collected automatically by means of software inventory tools. To link this information to known vulnerabilities, we rely on Common Vulnerabilities and Exposures (CVE), a well-established enumeration of publicly known cybersecurity vulnerabilities. We take advantage of our recent work on transforming this structured knowledge into a knowledge graph [14] available via various semantic endpoints. This allows us to directly integrate this information and use it in our scenario.

To implement the monitoring in this scenario, we set up a federated continuous SPARQL query at Listing 1.2 to identify whether a sensitive file shows up on a vulnerable workstation. To restrict the query to confidential files, we use the property `asset:hasDataClassification` and restrict our query to `sys:Private` files. Table 4 shows the query results and reveals that *Workstation2* and *Workstation3* have critical vulnerabilities, but store confidential files. The results include the `fileName`, `hostName`, `hostIP`, `cveId`, etc. As a next step, an analyst can inspect the life-cycle of the files to understand where they came from, who accessed them and explore information on the vulnerabilities and potential mitigations. Taking automated actions based on the results, such as blocking the access or alerting the user, is a further option.

6 Evaluation

In this section, we present our empirical evaluation setup and discuss the results.

[22] using a classification schema of confidential, private, protected, public

6.1 Experimental Setup

We ran the experiments on an Intel Core i7 processor with 2,70GHz, 16GB RAM, and 64-bit Microsoft Windows 10 Professional and emulate hosts as docker containers. We used C-Sprite as event extraction engine with a 3 seconds time window that slides every second. In order to simulate user activity, we developed a java-based event generator[23] to generate scripts for random file activities and use weighted random choices to select activities.

6.2 Experiments and Results

To measure the correctness and the completeness of the event extraction and detection using RDF stream processing with C-Sprite, we define a set of metrics, including *(i)* Actual Events (AE) – number of the events executed in the simulation (ground truth), and *(ii)* Returned Events (RE) – number of events correctly detected by the RDF-Stream processing (C-Sprite). We get detection (%D) by dividing RE by AE.

$$Detection(\%D) = \frac{ReturnedEvents(RE)}{ActualEventsGenerated(AE)} * 100\%$$

On each target OS (Linux and Windows), we test a varying number of events per second, i.e. 1, 10, 20, 50, 80, 100, 125 and 200 events/sec. In the results, we report the mean of detected events over 5 runs with 480 simulated events each.

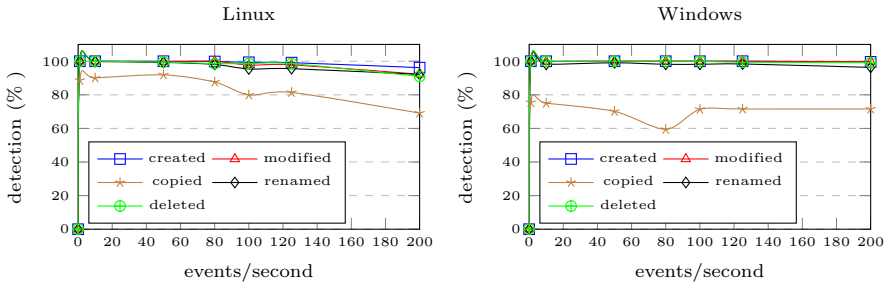

Fig. 5: Detection rate on Linux (l) and Windows (r)

As shown for Linux in Figure 5, all events can be detected close to 100% for all frequencies (1 event/sec up to 200 events/sec) except the copy event, which reached a maximum of 91,89%. At 200 events/sec, we observe that the detection of copy events decreases to approx. 70%, which is mainly caused by incorrect pairings of *readAttribute* and *create* events when these micro operations generated by two or more sequential copy events appear together in the same

[23] https://github.com/sepses/fileAccessExtractor/tree/master/eventGenerator

window. Furthermore, we noticed that low-level events sometimes do not arrive in sequence and hence, are not detected by our queries.

For Windows, the event detection performance for *created*, *modified*, *renamed* and *deleted* events is higher with almost 100% of detected events for all frequencies. However, the copy event detection in Windows achieves a lower detection with a maximum of 75,46%.

Finally, considering scalability we can make an estimation based on [5], which shows that C-Sprite achieves a throughput of more than 300000 triples/s. Consequently, it should be able to handle up to 23000 events/s (an individual event consists of at least 13 triples). For forensic scenarios, the Virtuoso triple store can load more than 500 million triples per 16GB RAM[24], which means that it should be possible to handle more than 38 million events per 16GB RAM.

7 Conclusions

In this paper, we tackled current challenges in file activity monitoring and analysis, such as the lack of interoperability, contextualization and uniform querying capability, by means of an architecture based on Semantic Web technologies. We introduced a set of vocabularies to model and harmonize heterogeneous file activity log sources and implemented a prototype. We illustrate how this prototype can monitor file system activities, trace file life cycles, and enrich them with information to understand their context (e.g., internal and external background knowledge). The integrated data can then be queried, visualized, and dynamically explored by security analysts, as well as be used to facilitate detection and alerting by utilizing stream processing engines.

Finally, we demonstrate the applicability of the approach in two scenarios in virtual environments – one focused on data exfiltration forensics, and another on monitoring policy violations integrating public vulnerability information. The results of our evaluation indicate that the approach can effectively extract and link micro-level operations of multiple operating systems and consolidate them in an integrated stream of semantically explicit file activities.

Overall, the results are promising and demonstrate how semantic technologies can enrich digital investigations and security monitoring processes. In future work, we aim to address the accuracy and scalability limitations of the current approach identified in the streaming evaluation, e.g., by evaluating alternative streaming engines and alternative approaches (e.g. complex event processing) based on big data technologies. Furthermore, we will investigate the integration of our approach into existing standards (e.g., STIX and CASE) to increase interoperability for forensic investigation.

Acknowledgments

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

[24] http://docs.openlinksw.com/virtuoso/virtuosofaq11/

Research Promotion Agency. The authors thank the funders for their generous support.

References

1. Alzaabi, M., Jones, A.: An ontology-based forensic analysis tool. In: Annual ADFSL Conference on Digital Forensics, Security and Law (2013)
2. Amato, F., Cozzolino, G., Mazzeo, A., Moscato, F.: An application of semantic techniques for forensic analysis. In: 32nd WAINA (2018)
3. Awad, A., Kadry, S., Maddodi, G., Gill, S., Lee, B.: Data leakage detection using system call provenance. In: International Conference on INCoS (2016)
4. Bhavsar, K., Trivedi, B.: Predicting insider threats by behavioural analysis using deep learning. In: International Conference on SAM (2018)
5. Bonte, P., Tommasini, R., De Turck, F., Ongenae, F., Valle, E.D.: C-sprite: Efficient hierarchical reasoning for rapid rdf stream processing. In: 13th ACM International Conference on DEBS. pp. 103–114. ACM (2019)
6. Carrier, B.D.: A Hypothesis-based Approach to Digital Forensic Investigations. Ph.D. thesis, Purdue University (2006)
7. Cheng, L., Liu, F., Yao, D.D.: Enterprise data breach: causes, challenges, prevention, and future directions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **7**(5) (2017)
8. Cuzzocrea, A., Pirró, G.: A semantic-web-technology-based framework for supporting knowledge-driven digital forensics. In: 8th MEDES Conference (2016)
9. Daren Fadolalkarim, E.B.: Pandde: Provenance-based anomaly detection of data exfiltration. Journal of Computer & Security **84**, 276–278 (2019)
10. Ekelhart, A., Kiesling, E., Kurniawan, K.: Taming the logs – vocabularies for semantic security analysis. In: 14th SEMANTiCS Conference (2018)
11. Gordon, P.: Data leakage - threats and mitigation. Report, SANS Institute (2007)
12. Hu, Y., Frank, C., Walden, J., Crawford, E., Kasturiratna, D.: Profiling file repository access patterns for identifying data exfiltration activities. In: IEEE Symposium on CICS (April 2011)
13. Kahvedžić, D., Kechadi, T.: Semantic modelling of digital forensic evidence. In: 2nd ICDF2C (2010)
14. Kiesling, E., Ekelhart, A., Kurniawan, K., Ekaputra, F.: The sepses knowledge graph: An integrated resource for cybersecurity. In: 18th ISWC (2019)
15. Kurniawan, K., Ekelhart, A., Kiesling, E., Froschl, A., Ekaputra, F.: Semantic integration and monitoring of file system activity. In: 15th SEMANTiCS (2019)
16. Mashwani, S.R., Khusro, S.: The design and development of a semantic file system ontology. Journal of Engineering, Technology & Applied Science Research **8** (2018)
17. Popitsch, N., Schandl, B.: Ad-hoc file sharing using linked data technologies. In: International Workshop on PSD 2010 (2010)
18. Schand, B., Popitsch, N.: Lifting file systems into the linked data cloud with tripfs. In: WWW2010 Workshop on Linked Data on the Web (2010)
19. Shen, Z., Hou, Y., Li, J.: Publishing distributed files as linked data. In: 8th International Conference on FSKD (2011)
20. Suresh, N.R., Malhotra, N., Kumar, R., Thanudas, B.: An integrated data exfiltration monitoring tool for a large organization with highly confidential data source. In: 4th CEEC (Sep 2012)