Graphameleon: Relational Learning and Anomaly Detection on Web Navigation Traces Captured as Knowledge Graphs

Lionel Tailhardat
Orange & EURECOM
Châtillon, France
lionel.tailhardat@orange.com

Benjamin Stach
Orange & UTBM
Belfort, France
benjaminstach.pro@gmail.com

Yoan Chabot
Orange
Belfort, France

Raphaël Troncy
EURECOM
Sophia Antipolis, France

ABSTRACT

User and Entity Behavior Analytics (UEBA) is key for managing security risks on information systems and comprehending user activities' impact on the network infrastructure. However, accessing network traffic and Web logs is challenging due to encryption or decentralized systems. Qualifying activities also requires contextualizing them according to the network's topology, as it determines potential exchanges and carries information about which services are used. This complexity hinders learning behavioral patterns when precise user action sequences are needed. We propose to tackle these challenges with Graphameleon, an open-source Web extension for capturing Web navigation traces. We model user activities in an RDF Knowledge Graph (KG), drawing from the UCO and NORIA-O ontologies. With this approach, we are able to distinguish analytics strategies implemented across different websites.

CCS CONCEPTS

• **Information systems → Web data description languages**; • **Computing methodologies → Knowledge representation and reasoning**; • **Security and privacy →** *Network security.*

KEYWORDS

Web Browsing Traces, User and Entity Behavior Analytics (UEBA), Knowledge Graph, Process Mining, Conformance Checking

ACM Reference Format:
Lionel Tailhardat, Benjamin Stach, Yoan Chabot, and Raphaël Troncy. 2024. Graphameleon: Relational Learning and Anomaly Detection on Web Navigation Traces Captured as Knowledge Graphs. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589335.3651447

1 INTRODUCTION

Behavioral analysis in cybersecurity is a technique that analyzes user, system, and network activities to identify and mitigate security threats. It aims to understand normal behavior patterns and detect anomalies that may indicate malicious activities. Understanding the precise impact of user activity on network behavior involves analyzing both the application logs and the network traffic. Unfortunately, application logs can be unusable due to their privacy or improper formatting. Similarly, network traffic can be encrypted or out of reach for collection, resulting in the loss of information about the user's interaction with the platform and the attack scenario [8]. In this work, we focus on the user-network-application system where interactions occur through a Web interface. We extend the concept of trace-based reasoning [3] to the domain of Web usage mining, considering the use of Knowledge Graphs (KGs) as a mean to consistently represent Web topology and usage data. More precisely, we address a new opportunity associated with the emergence of data models in the fields of network infrastructure (for the description of heterogeneous systems) and cybersecurity (for the description and management of attacks and risks). We assume that we can correlate descriptions of Web activities and usage with the description of the structure of the Web itself, in order to enhance the understanding and design of complex systems while considering the user-system pair. Our approach further extends Dynagraph, a system combining trace dumping tools with a Web app for rendering graph data from traces [12] to learn interpretable activity models in the form of Linked Data: the Graphameleon Web extension collects user activity traces (network traffic, interactions with the Web browser) during a Web navigation session and serializes this data in RDF using the UCO [18] vocabulary. The resulting data is then ingested into a KG to interpret the activity traces at a semantic level and derive patterns (Petri nets). These activity models could then be used to identify similar situations by projecting them onto the KG and, based on this projection, obtain contextual information by traversing the graph. The remainder of this paper is organized as follows. Section 2 presents some related work. In Section 3, we introduce our approach to capture knowledge from Web navigation traces based on a three-layer activity modeling and a data capture component. Section 4 describes our experiments and evaluations. Finally, Section 5 concludes the paper and discusses future work. We release the Graphameleon code at https://github.com/Orange-OpenSource/graphameleon and we publish the Graphameleon dataset at https://github.com/Orange-OpenSource/graphameleon-ds.

2 RELATED WORK

Web usage mining has gained significant attention for its practical insights into user behavior on the Web. Studies delve into diverse areas such as data preprocessing methods [15], user identification techniques [11], session recognition algorithms [5], and pattern discovery methods [6]. In activity modeling and analysis, trace-based reasoning [3] aids in creating tools for semantically interpreting digital services artifacts using controlled vocabularies and relational data models. The representation of events and activities within KGs is implemented across a range of data models, encompassing both domain-independent and domain-specific contexts: process modeling and execution (Petri nets-related [7], HTTPinRDF [9]); causal analysis (FARO [17]); cybersecurity (UCO [18], MITRE D3FEND [16]); network operations (NORIA-O [14]). It is evident that (particularly in UCO), the analysis of attacks and vulnerabilities is primarily based on indicators of compromise through the enumeration of artifacts from past situations. However, these indicators are never correlated with the topology of networks and services, or even the temporal structure between artifacts, which ultimately provides a static description of situations and overlooks the structure of activities (i.e. the strategy employed in its dynamics). In this regard, we notably showcase with our proposal how to incorporate the concept of navigation traces into UCO, thus simultaneously benefiting from cybersecurity knowledge and network context recorded elsewhere (i.e. by cybersecurity analysts and network operators) while ensuring a standardized representation of the data. Additionally, we extend the utilization of process mining [1] and conformance checking [10] to KGs, capitalizing on their alignment with the principles of trace-based reasoning.

3 APPROACH

To achieve data collection in such a way that one could analyze Web navigation traces within their network context or learn Web navigation activity templates, we propose a two-step approach. First, we utilize semantic modeling of user activities by leveraging a KG and UCO. Second, we develop a Web browser plug-in called Graphameleon, which captures data and serializes it in RDF.[1] Figure 1 provides an overview of our approach, illustrating the integration of Graphameleon in a processing pipeline. This pipeline aims to derive activity models, using process mining and Petri nets representation, for detecting Web navigation activity scenarios.

Semantic Modeling of User Activity. In the context of Web navigation, the concept of activity lacks a precise definition, as its interpretation heavily relies on the specific data and observation scale chosen. For example, one could analyze interactions during vehicle purchase on an e-commerce website or examine TCP packet exchanges between the client and server in detail. Let us first consider that an HTTP connection is established and a user-initiated request from the browser client is sent to the server hosting website. Most of the time, the requested document needs some additional resources like scripts, images, or other documents. Those dependencies will lead to a set of sub-requests. From our perspective, this action consists in navigating through a link in one single click (or accessing to the page through a URL), whereas from the Web browser it can be seen as a sequence of requests. In this paper, we consider this sequence as a so called "Micro-activity" trace. Then, exploring the level above, we could simply consider a trace as a set of requests and interactions. An interaction can be defined as any user-initiated action that has an impact on a webpage (click on link, form filling, etc.). We call such a trace a "Macro-activity". KGs help to homogenize data sources and address the challenge of interoperability. While the conventional functioning of Web browsers already relies on established standards and protocols, KGs can be beneficial when integrating data from sources outside of the Web browser context. We observe that UCO [18] appears to be well-suited as it enables the representation of Web navigation activities at various scales, including action cycles, individual actions, connections, protocols, resources, domains, and IP addresses. For the design of the mapping, the principle is to maximize the reuse of concepts/properties defined in UCO, and to match the fields and values captured at the Web browser level with these concepts/properties whenever their semantics align. An explanation of data model and the corresponding mapping rules in RML syntax [4] are available in our repository at https://github.com/Orange-OpenSource/graphameleon.

Data Collection with Graphameleon. We consider both the collection of HTTP requests and user/Web browser interactions to fully understand and analyze the user-network-application system. Graphameleon applies request listeners to both the sending and receiving processes at the background script level. This allows for the interception of all browser requests to retrieve information from the headers. This information includes URLs, associated IP addresses and domains, user-agent header, timestamp values, and fetch metadata request headers[2]. Fetch metadata allows us to derive indirect knowledge from navigation traces. For instance, the `Sec-Fetch-Site` indicates the relationship between a request initiator's origin and its target's origin, thus providing information about the network topology. Similarly, `Sec-Fetch-Mode` helps differentiate user-initiated requests from sub-requests for loading images and other resources. To abstract contextual elements and reduce diversity in activities, we tokenize the URLs of requested resources. This involves replacing arguments in the URLs with the names of their respective parameters. Graphameleon also injects content scripts into each of the Web browser's active tabs to capture interactions between the user and the browser. These scripts apply listeners to all interactive elements on the page, such as links, buttons, forms, etc. This approach reduces the impact on the browser performance and prevents capturing unwanted interactions, such as miss-clicks on non-interactive elements. To globally identify interactions, we consider the recorded event type, the element and the URL of the corresponding resource. When an element has an *id* attribute, its identification is straightforward. In many cases, webpage elements do not have identifiers, so it becomes necessary to identify an element based on its absolute position within the DOM hierarchy. This is done by calculating the absolute path of the element from the root of the DOM, resulting in an identifier

[1]We do not leverage the HTTP Archive (HAR) file format as it is incompatible with streaming and requires advanced user skills [12].

[2]https://www.w3.org/TR/fetch-metadata/

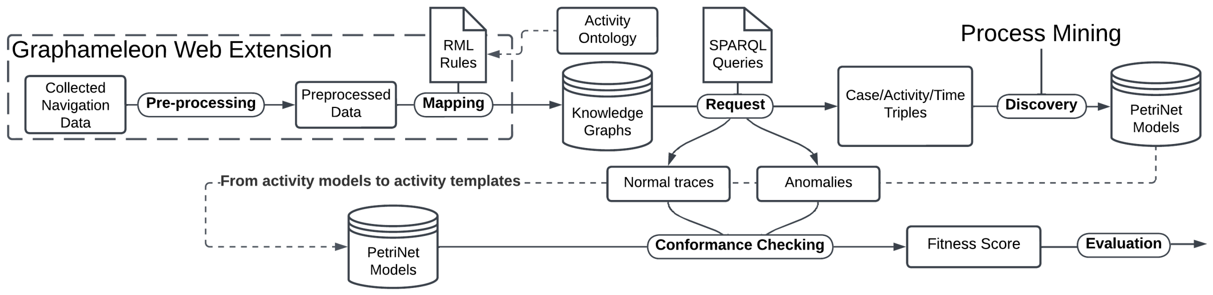

Figure 1: Overview of the data processing pipeline.
The Graphameleon Web extension captures and annotates user activity at the Web browser level. A process mining component derives activity models from the resulting RDF KG. These models can be used to build a library of activity templates, which are then used by a conformance checking component to classify new activity traces as normal or abnormal activities.

like `body > maindiv[2] > div > div > a`. While this method provides a deterministic way to identify elements without an *id* attribute, it is important to note that the resulting hierarchical path identifiers can be complex and difficult to interpret. Understanding which element is being referred to solely based on its path identifier is challenging without a capture of the webpage and the interaction collection for context analysis. One potential solution is to inject *id* attributes into webpage elements using listeners, but this may not address the issue of *id* stability between browsing sessions for pages with dynamic content.

4 EXPERIMENTS AND EVALUATION

This section describes two experiments conducted using the approach outlined in Section 3. We publish the dataset and provide additional technical details at https://github.com/Orange-OpenSource/graphameleon-ds.

Website complexity clustering. In this initial experiment, we seek to understand to what extent the behavior of a user visiting a website is crucial in creating a usable footprint subsequently leveraged for anomaly detection. For this, we focus on the ability to study the complexity of websites in terms of the number and the size of the resources to be loaded. We study this complexity through a Firefox desktop instance (anti-tracking $\in \{strict, standard\}$) by measuring the number of RDF triples generated by Graphameleon (collect mode $\in \{micro, macro\}$, output mode = *semantize*) during the initial connection to a website (the website's landing page), across a set of websites. Currently, there is no study describing well-known website complexity groups (clusters) but from a marketing perspective (e.g. industry sector *vs* average request count per landing page). We propose to use three categories based on the extent of editorial content being used: *One-Page*, *Encyclopedia* and *Content-Heavy*. We select three reference websites, one for each category based on third-party expert opinions.

Using this setup, 27 data collections were performed (three categories × three sites × three data collection setups). For data collection in *macro* mode, we observe that the number of RDF triples remain consistent regardless of the visited website. Regarding the *micro* mode, the counts for a given anti-tracking policy configuration exhibit significant variability within each complexity category. In this line of thought, Table 1 focuses on the mean entity count for the `ucobs:HTTPConnectionFacet` (UHC)

	Strict		**Standard**		**Std. / Str.**	
	UHC	UIP	UHC	UIP	UHC	UIP
One-Page	61.0	4.0	63.5	7.0	1.04	1.8
Encyclopedia	46.7	6.3	127.7	41.7	**2.73**	**6.6**
Content-Heavy	37.0	6.3	60.7	14.7	1.64	2.3

Table 1: Average number of entities in micro mode.
Comparison of the average UHC and UIP entities count as a function of the complexity level and of the anti-tracking policy. The following abbreviations apply: UHC = *ucobs:HTTPConnectionFacet* entities count, UIP = *ucobs:IPAddressFacet* entities count.

and `ucobs:IPAddressFacet` (UIP) object classes, and for each scenario. The comparison of the mean entity count values based on the anti-tracking policy ("Std. / Str." column in Table 1) reveals an increase in the average number of connections and remote servers accessed when the anti-tracking rules are relaxed, regardless of the complexity level. Based on these measurements, we conclude on the correct functioning of Graphameleon and its suitability for studying initial connection behaviors. While the current proposed complexity categories might not be pertinent for website grouping due to limited sample size and content variability, the rise in network exchanges under varied anti-tracking policies provides a basis for future categorization by employed analytics strategies and network topology.

Navigation trace classification. In this second experiment, our goal is to classify Web navigation traces as either normal or abnormal behaviors. Using macro-activity modeling (Section 3), process mining [1] and conformance checking [10], we analyze the following three scenarios. **Base scenario (normal behavior):** a user accesses the website, logs in with a username and a password, navigates to the "Sell a Book" page, fills out the form, and then returns to the homepage in order to find the book. **Alternative scenario (different behavior):** same as the Base scenario but uses a Single Sign-On (SSO) for the authentication. **XSS attack scenario (abnormal behavior):** same as the Base scenario with a code injection in the "Author" field of the "Sell a Book" page and returning to the homepage where the injected script is executed.

We deployed a simulated online bookstore website in order to maintain control over this experiment. We proceed with the data collection of navigation traces data for each scenario with Graphameleon (collect mode = *macro*, output mode =

semantize) in a Firefox desktop instance, then compute the activity model from the saved trace using the PM4PY Process Mining library [2] (process discovery $\in$ $\{Inductive, Alpha, LogSkeleton, Heuristic, AlphaPlus\}$), and finally compute the fitness of the activity model against the activity template by also using PM4PY (process mining $\in$ $\{TokenBasedReplay, Alignment\}$). The base scenario, which corresponds to the behavior we define as "normal", is used to create our reference model (activity template).

	Base	**Alternative**	**XSS Attack**
Request	10	13	11
Interaction	18	14	18
Vertice	263	283	277
Edge	404	431	426

Table 2: Data collection statistics for the navigation trace classification experiment.
Statistics from the Graphameleon user interface in terms of the number of network requests, user interactions with the Web browser, nodes and edges of the resulting navigation graph.

Three data collections were conducted using this procedure. Table 2 summarizes the statistics for the resulting navigation graph. From a graph statistics perspective and with reference to the "base" scenario, we observe that the "alternative" scenario involves fewer interactions but more network transactions. This is because authentication is streamlined with a single button click and delegated to external entities. In the "XSS attack" scenario, interactions remain constant, but requests increase by one due to an additional query involved by the SQL injection step. Still for this same scenario, we notice a slight variation in the fitness scores (average of 98% fitness, calculated with the "normal" activity model as reference), which also corresponds to the single additional request caused by the SQL injection step. We further observe that this additional query is easily identifiable through sequence alignment using the developed vocabulary applied at the level of the Web extension to standardize the interpretation of the traces. Reflecting on data collection and semantic processing, we find that there is minimal lexical compression of navigation trace data due to consistent formatting (e.g. the request URL is consistently located using the "url" header). However, this compression pertains more to interaction semantics. Indeed, one challenge in aligning activity models stems from the lack of a reliable method to identify HTML elements (especially when lacking an explicit *id*) across browsers, sessions, and users. This challenge becomes apparent when the DOM of page content changes with each site visit, especially when dynamic ad insertions occur.

5 CONCLUSION AND FUTURE WORK

In this work, we considered the combined use of KGs and Web browser-level navigation trace collection as a means to capture network behavior details and assist in modeling cyberattacks. This led to the development of the concepts of micro-activity and macro-activity for semantic representation of user activities. We also created Graphameleon, an open-source Web extension available at https://github.com/Orange-OpenSource/graphameleon for live data collection and semantization of navigation traces at the Web browser level. Analyzing the activity traces collected by this component in various websites, we showed that the rise in network exchanges under varied anti-tracking policies provides a basis for future categorization by employed analytics strategies and network topology. Through a complementary navigation trace classification experiment, we showed how activity models could be derived and analyzed leveraging Petri nets and conformance checking technique. We also highlighted the challenge in aligning activity models from the lack of a reliable method to identify HTML elements across browsers, sessions, and users. Future work will focus on Web cartography, behavior analytics, and anomaly detection in an adversarial setting. This will include incorporating activity models as part of anomaly context data for decision support applications using graph embeddings [13]. Privacy concerns will also be explored in order to evaluate to what extent the overall approach enables the achievement of specific detection cases.

REFERENCES

[1] Wil Aalst, A. Weijters, and Laura Măruşter. 2004. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering* (2004).

[2] Alessandro Berti, Sebastiaan van Zelst, and Wil van der Aalst. 2019. Process Mining for Python (pm4py): Bridging the Gap between Process-and Data Science. In *Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019).*

[3] Amélie Cordier, Marie Lefevre, Pierre-Antoine Champin, Olivier Georgeon, and Alain Mille. 2013. Trace-Based Reasoning - Modeling Interaction Traces for Reasoning on Experiences. In *The 26th International FLAIRS Conference.*

[4] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Proceedings of the Workshop on Linked Data on the Web, LDOW 2014, co-located with the 23^{rd} International World Wide Web Conference (WWW 2014).* CEUR-WS.org.

[5] Maria Carla Calzarossa and Luisa Massari. 2014. Analysis of Header Usage Patterns of HTTP Request Messages. In *IEEE International Conference on High Performance Computing and Communication.*

[6] Giovanna Castellano, Anna M. Fanelli, and Maria A. Torsello. 2013. *Web Usage Mining: Discovering Usage Patterns for Web Applications.* Springer Berlin Heidelberg, 75–104.

[7] Dragan Gašević and Vladan Devedžić. 2006. Petri Net Ontology. *Knowledge-Based Systems* (2006).

[8] Iman Akbari, Mohammad A. Salahuddin, Leni Ven, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stephanie Moteau, and Stephane Tuffin. 2022. Traffic Classification in an Increasingly Encrypted Web. *Commun. ACM* (2022).

[9] Johannes Koch, Carlos A. Velasco, and Philip Ackermann. 2017. *HTTP Vocabulary in RDF 1.0.* W3C Working Group Note. W3C. https://www.w3.org/TR/HTTP-in-RDF10

[10] Jorge Munoz-Gama. 2014. *Conformance Checking and Diagnosis in Process Mining: Comparing Observed and Modeled Processes.* Ph. D. Dissertation. Universitat Polit'ecnica de Catalunya – BarcelonaTech, Barcelona.

[11] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. 2019. Browser Fingerprinting: A survey. arXiv:1905.01051

[12] Lionel Tailhardat, Raphaël Troncy, and Yoan Chabot. 2022. Walks in Cyberspace: Towards Better Web Browsing and Network Activity Analysis with 3D Live Graph Rendering. Association for Computing Machinery. https://doi.org/10.1145/3487553.3524230

[13] Lionel Tailhardat, Raphael Troncy, and Yoan Chabot. 2023. Leveraging Knowledge Graphs For Classifying Incident Situations in ICT Systems. In *18th International Conference on Availability, Reliability and Security (ARES).*

[14] Lionel Tailhardat, Yoan Chabot, and Raphaël Troncy. 2024. NORIA-O: an Ontology for Anomaly Detection and Incident Management in ICT Systems. In *21^{st} Extended Semantic Web Conference (ESWC).*

[15] Vítor Santos Lopes and João Mendes-Moreira. 2019. A Comparative Analysis of Data Preprocessing Techniques in Web Usage Mining. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI).* IEEE, 134–139.

[16] Peter E. Kaloroumakis and Michael J. Smith. 2021. *Toward a Knowledge Graph of Cybersecurity Countermeasures.* Technical Report. The MITRE Corporation.

[17] Youssra Rebboud, Pasquale Lisena, and Raphael Troncy. 2022. Beyond Causality: Representing Event Relations in Knowledge Graphs. In *Knowledge Engineering and Knowledge Management.* Springer International.

[18] Zareen Syed, Ankur Padia, M. Lisa Mathews, Tim Finin, and Anupam Joshi. 2016. UCO: A Unified Cybersecurity Ontology. In *AAAI Workshop on Artificial Intelligence for Cyber Security.* AAAI Press.