Unveiling Attack Patterns from CTF Network Logs with Process Mining Techniques[⋆]

Francesco Romeo[1,2], Francesco Blefari[1,2], Francesco A. Pironti[1,†] and Angelo Furfaro[1,*]

[1] *DIMES – University of Calabria, P. Bucci, 41C, 87036, Rende, CS, Italy*
[2] *IMT School for Advanced Studies, Piazza S. Francesco, 19, 55100 Lucca, LU, Italy*

Abstract

Providing concise yet sufficiently detailed representations of potentially malicious interactions with network-exposed services could simplify attack analysis and help identify exploited vulnerabilities. A key challenge is the automatic, real-time generation of such representations to enable prompt defensive actions and rapid responses. Capture the Flag (CTF) competitions of type Attack and Defense (AD) are a gamified example of scenarios where the availability of such a tool could play a critical role. In addition, in CTF-AD environments, the problem of reverse engineering from network packet to high-level interactions is exacerbated by some provisions used to hide the packets' source.

This paper proposes an effective solution, based on process mining techniques, which is able to identify and infer the attacker's behavior and to produce its representation as a Directly-Follows Graph (DFG). The approach has been thoroughly validated by exploiting a Cyber Range scenario where N teams fight a CTF-AD competition, comprising: a game server, a set of N machines, one for each team, hosting vulnerable services and from where the own services are handled, and a set of $M \geq N$ machines, one for each simulated player, from where attacks are launched. The developed tool can be used by teams to analyze attacks on their services in order to identify exploited vulnerabilities and replicate them against adversaries.

Keywords

Log-Analysis, Process Mining, Cyber Range, Network Log,

1. Introduction

The ever increasing pervasiveness of connected devices and applications in people's everyday lives has improved the way many tasks, even critical or hazardous ones, are carried out. However, at the same time, the potential interferences due to the exposure to cyber threats are becoming a very important issue that still needs to be adequately taken into account. Healthcare, infrastructure monitoring, economic transactions, energy management and other critical applications are vulnerable to Internet threats and, as highlighted in [1, 2], it is necessary to ensure security in this crucial contexts. The process of guaranteeing confidentiality, integrity, and availability of IT systems is further complicated by the continuous emergence of new threats and by the heterogeneity of the involved devices. Elaborating an effective countermeasure against cyber-attacks is not a trivial task to carry out. The difficulties are increased by attacks exploiting 0-day vulnerabilities. In this context, the availability of tools able to automatically detect and analyze anomalous behaviors may play an important role [3].

Detection and analysis of security threats are still performed largely manually and often use highly specialized techniques tailored for the specific domain. Modelling-based techniques are employed to

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT

[⋆] This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

[*] Corresponding author.

[†] The work of Francesco A. Pironti was supported by Agenzia per la cybersicurezza nazionale under the 2024-2025 funding programme for promotion of XL cycle PhD research in cybersecurity (CUP H23C24000640005). The views expressed are those of the authors and do not necessarily represent the funding institution.

francesco.romeo@unical.it (F. Romeo); francesco.blefari@unical.it (F. Blefari); francesco.pironti@unical.it (F. A. Pironti); angelo.furfaro@unical.it (A. Furfaro)

0009-0006-3402-3675 (F. Romeo); 0009-0000-2625-631X (F. Blefari); 0009-0003-3183-2977 (F. A. Pironti); 0000-0003-2537-8918 (A. Furfaro)

infer behavioral models of the involved entities, encoded in a mathematical notation [1]. However, these techniques, if not properly used, may produce models that could be very abstract, too complex, and hard to manipulate. Some attack modeling techniques known in literature are: (a) Attack Surface [4]; (b) Kill Chain [5]; (c) OWASP's threat model [6]; (d) Diamond model [7]; (e) Attack Vector [8]; (f) Attack Tree or Graph [9, 10].

This paper proposes an attack modeling approach, based on process mining techniques [11], which can be exploited to infer the process model corresponding to the behavior of an external entity, interacting with a given system, whose behavior is deemed as being anomalous and then probably malicious. Process models of detected malware can be used to understand the attacker's cyber-kill chain and identify the exploited vulnerabilities (e.g. zero-day). The effectiveness of the approach has been evaluated in the context of Capture the Flag (CTF) competition of type Attack and Defense (AD) executed into a Cyber Range (CR) platform.

This paper proposes an effective solution, based on process mining techniques, which is able to identify and infer the attacker's behavior and to produce its representation as a Directly-Follows Graph (DFG).

The remainder of the paper is organized as follows. Section 2 introduces the background giving a short overview about process mining, Cyber Ranges and CTF environments. Section 3 illustrates the proposed methodology. A Proof-of-concept implementing the proposed approach and an evaluation is given in section 4. Section 5 concludes the paper and discusses future works.

2. Background

2.1. Process Mining

The aim of Process Mining-based techniques is to extract the knowledge held by a process in such a way to provide broader perspectives for the interpretation of the data, with the goal of obtaining a better understanding of the process behavior. By leveraging process mining techniques, it is possible to discover, monitor, and improve real processes by extracting knowledge from event logs. Clearly, process mining is particularly relevant in a context where the involved actors are autonomous and can deviate or exhibit emergent behaviors. The more ways in which services, people, and organizations can deviate, the more interesting it is to observe and analyze the corresponding processes as they are executed. Three fundamental types of process mining tasks can be identified [12]:

- *Discovery*: the model is constructed based on an event log. In this case, there is no a priori model.
- *Conformance*: in this case, an a priori model is used to verify if reality conforms to the model. It is employed to detect, identify, explain, and measure the severity of the discovered deviations.
- *Extension*: similarly to the conformance case, there exists an a priori model, but it is extended with a new aspect or perspective. The goal here is not to verify conformity but to enrich the model.

From the cyber security perspective, the use of PM techniques relying on process discovery are effective in that, unlike classical monitoring solutions, they allow to reduce implementation time by not requiring the availability of a process model in advance. Moreover, there are some approaches which combine classical data-centric (e.g. Data Mining) approaches and process-centric approaches (e.g. BPM Analysis) to give better results in addressing security issues [13]. These approaches represent an added value since they are designed to examine when and how an actual process deviates from a given process model, thus giving critical insights on the real-time behavior of a system, for example deviation from the *normal* evolution toward a (potential) unsafe state.

As stated before, the aim of process discovery is to infer a process model, expressed in a suitable mathematical notation (e.g., Petri Net, BPMN, process tree), from the analysis of a set of traces [14]. The obtained model must be able to explain all the recorded logs. The resulting "mined" model can be employed in a workflow management system or used to perform analysis on the actual process behavior, thus identifying deviations between the model and the recorded behavior. A process discovery algorithm usually performs the following subtasks: *(i)* analyzes traces or logs; *(ii)* extracts causal dependencies; *(iii)* presents results in the form of a dependency graph; *(iv)* enriches the resulting dependency graphs with advanced aspects of process behaviors by returning a formalized process model in specific process modelling language. From a given log trace, different process models can be extracted, and a choice of the one considered the *most appropriate* has to be made. Process discovery is often carried out using heuristic approaches, as can be seen in [15], and the quality of the resulting models increases proportionally with the number of available traces. The data set size has to be adequate, indeed, if it is too small, the quality of the models tends to be very poor.

Process mining tools such as *ProM* [16] are capable of working with considerable amounts of data, allowing process mining to be applied to real web services without any problem.

2.1.1. Directly-Follows Graph

Directly-Follows Graphs [17] (DFGs) are commonly used in process mining to explore event data. A trace, or process variant, $\sigma = \langle a_1, a_2, \dots, a_n \rangle$, is a sequence of activities. $\#_L(\sigma)$ denotes the number of cases in the event log L that correspond to σ, while $\#_L(a)$ and $\#_L(a, b)$ represent the occurrences of activity a and the directly-follows relation (a, b), respectively. Without loss of generality, it is assumed that each case starts with a start event (▶) and ends with an end event (■). A DFG is a directed graph where nodes represent activities, and edges represent directly-follows relationships. Three thresholds, τ_{var}, τ_{act}, and τ_{df}, filter out infrequent traces, activities, and relations, respectively. The construction process is as follows:
1. *Input:* Event log L and thresholds τ_{var}, τ_{act}, τ_{df}.
2. *Filter traces:* Remove traces σ with $\#_L(\sigma) < \tau_{\text{var}}$, producing L'.
3. *Filter activities:* Remove activities a with $\#_{L'}(a) < \tau_{\text{act}}$, resulting in L''.
4. *Add nodes:* Create a node for each activity in L''.
5. *Add edges:* Connect nodes a and b if $\#_{L''}(a, b) \geq \tau_{\text{df}}$.
6. *Output:* A DFG with nodes labeled $\#_{L''}(a)$ and edges labeled by $\#_{L''}(a, b)$.

Additional timing statistics can be computed for each edge.

This process results in a simplified DFG that focuses on significant activities and relationships in the event log. Some limitations of DFGs are their susceptibility to producing misleading results due to improper filtering or threshold adjustments, which can lead to distorted activity sequences and inaccurate performance diagnostics. Furthermore, DFGs lack the expressiveness to represent complex process structures such as parallelism, loops, or routing logic.

Despite these weaknesses, DFGs present advantages over other types of representations in terms of immediacy of comprehension. Therefore, DFGs represent a good choice in a scenario where having an easy to understand representations is deemed more important than obtaining extremely accurate representations of complex structures.

2.1.2. Anomaly Detection Process Model

Due to the limitations of manual analysis, there is a growing demand among developers for automated log analysis methods. During the past decade, extensive research has focused on innovative approaches to automatic log analysis and anomaly detection [18, 19, 20], resulting in the development of a four-phase process: *(i)* log collection; *(ii)* log parsing; *(iii)* feature extraction; *(iv)* anomaly detection [21].

Usually, in the log collection phase, the logs are retrieved from their storage locations for analysis. Each log must present a timestamp and a descriptive message indicating its parent event, as well as other descriptive fields. The collected logs are then processed in order to detect anomalies. Starting from the parsing phase, event models are extracted from previously collected unstructured logs to obtain structured logs. These logs are composed of static and dynamic components. The two main approaches to carry out log analysis are: • *cluster-based*: it utilizes distances between logs for clustering and event template generation; and • *heuristic-based*: it identifies frequent words to compose candidate log events. In the feature extraction phase, the structured logs are parsed into numerical feature vectors and categorized into sequences using three grouping techniques: fixed windows, sliding windows, and session windows, each based on different criteria. This is a fundamental phase which transforms logs to enable the application of machine learning techniques. A feature vector is generated for each log sequence that represents the occurrence number of each event. The fourth phase is the detection of anomalies. The output of this phase is a feature matrix which can be provided to a machine learning model for training and generation of an anomaly detection model. Based on the inferred model, a new incoming log sequence can be classified as anomalous or normal.

2.2. Cyber Range Platforms

The growing number of attacks occurred in recent years, raises constantly the need of highly skilled professionals. There is the necessity of cyber-infrastructures able to support training, vulnerability assessment, and testing activities. Cyber Ranges (CRs) platforms, which are composed of several components and have characteristics that allow them to be able to adapt to a variety of different scenarios, play an important role in this panorama. It is possible to define a Cyber Range as an exercising environment containing both physical and virtual components to represent realistic training scenarios [22]. With this need in mind, Capture the Flag (CTF) or Attack and Defense (AD) challenges can be seen as a way to improve the player's skillset.

The National Institute of Standard Technology (NIST) highlighted that cyber security is a nowadays challenge requiring even more skilled experts [23]. Indeed, today's cyber security attacks are increasingly complex, often coalescing vulnerabilities having different nature. In [23], *Cyber Range* (CR) platforms are identified as a valid tool useful in reducing the lack of knowledge in the field of cyber security. By supporting this assertion, over the years, CR platforms became one of the most promising advances in the cyber security area for testing and training purposes. Despite the fact that solid theoretical security knowledge constitutes the foundation to face cyber threats and vulnerabilities, cyber security experts need to regularly train their abilities in order to enhance their skills and improve their expertise. Using CRs, it is possible to provide a suitable platform capable of replicating real world scenarios [24, 25, 26] by including: *(i)* physical and virtual components [22, 27]; *(ii)* software components able to emulate/simulate some (physical) systems' behavior; *(iii)* entities having specific software vulnerabilities [28] which are under study.

According to the state-of-the-art literature and to the NIST specification, CRs are defined as "interactive and simulated representations pertaining to an organization's local network, system, tools, and applications" [29]. CRs are widely adopted in cyber security field, indeed, they are commonly used to emulate (or simulate) specific computer environments, especially to test system or infrastructure security, and to train people across different cyber security fields [30, 31]. The CRs platform acts as a virtual playground for cyber security professionals, students and researchers, becoming the base technology to assess and improve their knowledge and practice against cyber threats. The realistic and controlled environment provided by a CR allows to create specialized purposes scenarios which mimic real-world assets (networks, systems and applications) including simulation of auxiliary infrastructures (such as servers, clients, firewalls and routers). The main advantage offered by CRs is the high isolation level that these platforms ensure. Examples of useful scenarios supported by CRs are: *(i)* red team vs blue team battlefield; *(ii)* cyber security training environment for certification [32]; *(iii)* incident response, forensics investigation, penetration testing, and vulnerability assessment scenarios for testing purposes; *(iv)* malware analysis or reverse engineering scenarios for research purposes [33]. According to the authors of [30], the creation of CRs is a challenging task, thus they propose the "Cyber Range Instantiation System (CyRIS)" in order to ease the creation of complex CR Platforms. A new development approach of next-gen CRs systems, based on cloud computing and hypervisor technologies, that could improve the realism leveraging statistical and AI techniques has been recently proposed in [27].

2.3. Attack & Defense Capture the Flag

A particular type of red team vs. blue team operations in a Cyber Range environment is an “Attack and Defense” (A/D) Capture the Flag challenge game. These games are designed as a learning tool for enhancing participants’ skills and as test bed for new platforms, attacks, and attacks and/or defense methodologies that will be employed in the wild. In each challenge, every team has to defend a machine (*vulnbox*). This specific scenario can be used for both training and fun purposes. An A/D game consists of several rounds of equal duration. Within each round, the game server interfaces with the *vulnbox* assigned to each team in order to refresh the stored *flags* by employing a series of scripts called checkers. In this context, the *flag* term, refers to a secret information that each team has to steal from the others. Once a *flag* has been successfully obtained, it is sent to the game server in order to redeem points. In the same way, when an attacker team steals a *flag* and submits it, the defenders lose points. At the end of the competition, the team with the highest score wins the challenge.

Each *vulnbox* has a number of exposed services and the machine owner’s team has to keep all services active in order to maintain their Service Levels Agreement (SLA) as high as possible. The entire game is handled by a component named *Game Server*. It is the core component which hosts some services for each team: *(i)* a scoreboard allows to see the current point distribution and the SLA for each service; *(ii)* a service known as *FlagID* is exposed in order to communicate some bit of information needed to find the flags (e.g the name of the right user); *(iii)* another services is responsible to authenticate each team and accept or deny the provided flags. If a specific service does not work properly or is not reachable, a penalty is assigned to the owner team. Such an operation is made by the Game Server that checks also: *(i)* the availability of all services updating the SLA score accordingly; *(ii)* if it is possible to insert a new flag; *(iii)* if it is possible to retrieve the previously inserted flag. Teams’ machines and Game Server, cannot be identified by their IP address due to the Network Address Translation (NAT), which hides the real IP addresses in order to avoid identification of attackers.

3. Methodology

The devised approach is able to perform process mining on logs coming from hosts deployed in a CR environment and has been tailored to work with HTTP-based web challenges. In this paper, we decided to limit the evaluation of the methodology to the above-described contexts in order to have a functional proof-of-concept that can be easily tested in real life challenges. We plan to expand the validation to more general settings in future work. The basis of the proposal relays on customizing the anomaly detection process in order to infer a model describing events occurring into the analyzed service. Indeed, differently from the classic anomaly detection process, we replaced *(i)* the feature extraction phase with a custom-made phase to pre-process log data, and *(ii)* the anomaly detection process phase, with the visualization of a dependency graph in such a formalism to better understand dependencies among network events that occurred into the service. The resulting four-phase process is: *(i)* Log collection; *(ii)* Log parsing; *(iii)* Log preprocessing; *(iv)* Inferred Model Visualization. During the log collection phase, basic network packet sniffers are employed to gather logs. The log parsing is responsible for converting them into a more structured and lightweight JavaScript Object Notation (JSON) representation, stripped of unnecessary data. In the Log preprocessing phase, parsed logs are computed by a packet aggregator based on domain-specific metrics in order to reconstruct different HTTP sessions. In the inferred model visualization phase, organized logs are refined by applying a particularly suitable process mining algorithm in order to produce a better representation of the events. In this phase we (i) perform process discovery analysis, and (ii) visualize the related process graph.

Log collection. The log collection phase is carried out by using a packet sniffer tool (such as tcpdump [34]). Such a tool, intercepts all network traffic for a specific web service storing it into “pcap” files to extract useful information regarding network ISO/OSI levels from 2 to 7 in the following phase.

Log parsing. Each network packet, in the “pcap” file, is analyzed in this phase that performs their conversion following the JSON specification. Moreover, pre-filtering, retaining only relevant fields according to Table 1 for the HTTP request packets and for HTTP response packets, is performed. The output of the log parsing phase is a list of JSON objects containing processed network packets.

Field	Description
`timestamp`	The absolute timestamp
`src_ip`	The source IP address
`src_port`	The source port
`dst_ip`	The destination IP address
`dst_port`	The destination port
`packet_type`	Specifies if such a packet is a request or a response
`HTTP_method`	The HTTP method used
`request_uri`	The URI
`status_code`	The status code related to the HTTP response (Only in HTTP response)
`cookie`	The cookie
`body`	The body content

Table 1
HTTP request/response fields

Log preprocessing. In this phase, the JSON list is transformed in order to obtain more structured and useful data suitable for process mining algorithms. The *Log Preprocessing* phase is composed of three software modules that sequentially handle data, respectively: *(i)* to reconstruct data, *(ii)* to filter and enrich data, *(iii)* to convert the JSON list according to the eXtensible Event Stream (XES) format, an XML-based format commonly used by process mining algorithms. The first module, called *Session Reconstructor*, has been designed to merge related network flows based on *(i)* TCP connection, and *(ii)* cookie field or authentication field. The idea behind session reconstruction is to identify all TCP sessions among all TCP flows in order to merge multiple TCP sessions based on the cookie or authentication fields present into the request (or response) header. The second module, called *Log filtering and marking*, initially performs a packet filtering, retaining only relevant HTTP packets. Subsequently, a string describing the activity is added to each entry of the filtered logs in the previous module. If the log entry is an HTTP request, the activity description is composed of *HTTP_method* and *request_uri*. On the other hand, in the case of an HTTP response, the activity description is obtained as *status_code*, *HTTP_method*, and *request_uri*. Lastly, reconstructed sessions are analyzed using a pattern matcher to spot leaked sensitive information. If an information leakage is detected, a marker is added to the whole session. Finally, the module *Parser Dict to XES* is used to translate pre-processed log data into the XES format.
Inferred Model Visualization. The inferred model visualization phase is carried out using process mining algorithms. In this phase, the XES file received as input is processed by a specific process mining algorithm to: *(i)* create causal dependencies among all information (representing network packets) present in each trace, and *(ii)* to obtain a model (visualizable in such a formalism) that specifies attackers' interaction with the analyzed system.

4. Proof-of-concept and evaluation

To prove the effectiveness of our methodology, we designed a proof-of-concept architecture. The hardware employed in our experiments is based on a server HP ProLiant DL360 gen8 featured by a 6x 8-core Intel Xeon E3-1260 processor, 256 GB of ram and 6 Solid State Drive (ssd). This server is equipped with Proxmox VE [35], a type 1 hypervisor that facilitates the virtualization of virtual machines running Unix or Windows operating systems and LXC containers. In choosing a scenario suitable for the proposed approach where it can truly be audited and useful but also validated, the Attack and Defense CTF challenges provide a good testing ground. Moreover, this challenges mimics real-world vulnerabilities (e.g. services exposing SQL injection vulnerabilities) in a fast paced environment where

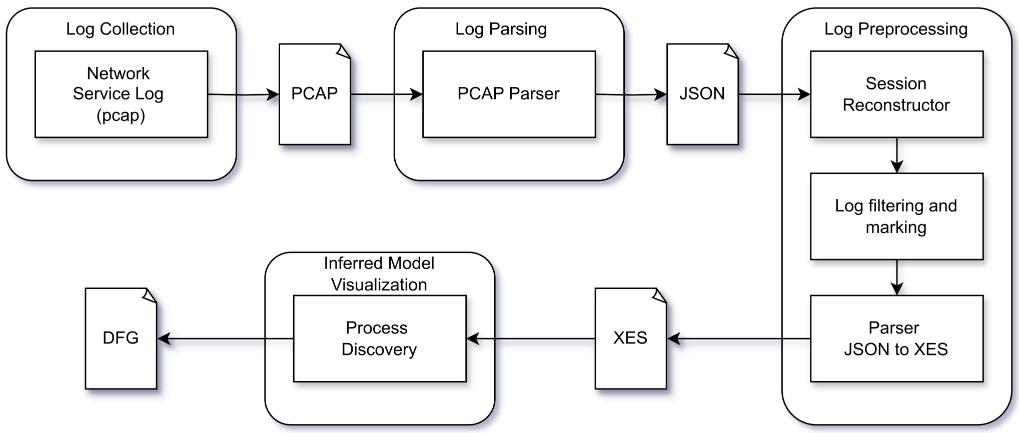

Figure 1: The proposed methodology for network log analysis

every minute count. Having a tool that can help to understand the opponent's attacks leads to faster patch and overall better scores, giving the users of this tool an advantage. The rigorous structure of the flag allows treating this particular data as a marker, enabling a better understanding of what an anomaly is (e.g. Unauthorized flag inclusion in a file) and the steps that the attackers has taken to gain that flag.

4.1. Methodology implementation steps

For each phase of our methodology, we developed ad hoc software modules that, when computing input data, provide output data suited for the next module or visualization (in the case of the final phase). The first phase collects network logs from a specific service. To this aim, we used tcpdump [34], a packet sniffer based on eBPF technology [36]. The output of this phase, as described in Section 3, is a "pcap" file that contains all captured network packets; then it is served as input of the *Log Parsing* phase to be converted using the *PCAP Parser* module. To this end, we relied on `PyShark` [37], a Python wrapper for the tshark network protocol analyzer [38], allowing python packet parsing using wireshark [38] dissectors. To perform process mining analysis, we used the `pm4py` Python library, which provides process discovery algorithms. Since all process discovery algorithms in the `pm4py` library take in input the data according to the `XES` format, we developed a conversion module to convert `JSON` data to meet `XES` specification. In our proof-of-concept, we used only the inductive miner as process discovery algorithm even if we designed the whole system to accommodate the addition of other process mining algorithms.

`pm4py` also allows the use of thresholds in the form of percentage. A proper setting of the threshold values is crucial for the effectiveness of the mining process. A low threshold results in a highly detailed graph, which could contain noise. On the other hand, a high threshold results in a loss of relevant information. For this reason, after empirical evaluations, we choose to use only path thresholds with a value of 30%.

4.2. The Cyber Range platform

For proof-of-concept purposes, hosting a complete game infrastructure on a cyber range was deemed too complex. Instead, a reduced architecture was instantiated. To demonstrate the effectiveness of the proposed approach, a small-scale cyber range was created and populated with a few vulnboxes.

The scenario was configured to support a game setup that involves one vulnerable service exposed. In this environment, three containers based on Ubuntu 20.04 were deployed: the first container (with IP address `10.60.0.1`) is the vulnbox where the network logs are collected and the services are exposed. Another container (`10.60.0.100`) is simulating an attacker where a custom made script continuously launches attacks directed to the vulnbox. The third machine (`10.60.0.254`) is configured as a Game

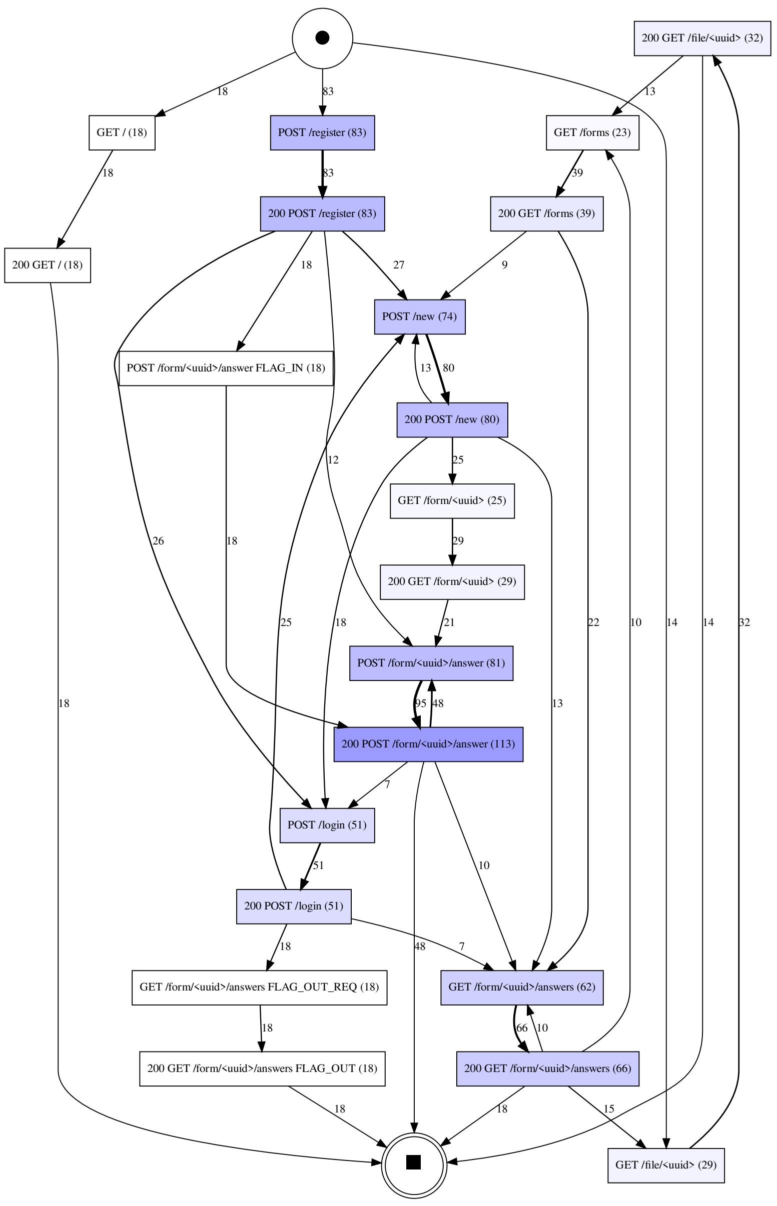
200 GET /file/<uuid> (32)
GET / (18)
POST /register (83)
GET /forms (23)
200 POST /register (83)
200 GET /forms (39)
200 GET / (18)
POST /new (74)
POST /form/<uuid>/answer FLAG_IN (18)
200 POST /new (80)
GET /form/<uuid> (25)
200 GET /form/<uuid> (29)
POST /form/<uuid>/answer (81)
200 POST /form/<uuid>/answer (113)
POST /login (51)
200 POST /login (51)
GET /form/<uuid>/answers FLAG_OUT_REQ (18)
GET /form/<uuid>/answers (62)
200 GET /form/<uuid>/answers FLAG_OUT (18)
200 GET /form/<uuid>/answers (66)
GET /file/<uuid> (29)

Figure 2: CCForms Challenge - Checkers graph. The labels on nodes and edges represent, respectively, the occurrences of activities and directly-follows relations.

Server continuously running the checkers scripts and exposing a custom-made flagID service. Since all services run within the same network (10.60.0.0/24), a virtual switch connecting the three containers suffices. For more complex scenarios, virtual routers could be introduced to enhance the network topology.

4.3. Challenge “CCForms"

The CCForms service is a website where it is possible to create forms with multiple questions and collect relative answers. It is a Webapp with a React front-end and a NodeJS back-end. A functionality of this application is the possibility of writing a public or private note. Although CCForms has multiple vulnerabilities, the one chosen for this experiment is an Insecure Direct Object References (*IDOR*) vulnerability allowing unauthenticated agents to request directly a page and get the flag. This challenge was proposed in the 2024 Attack and Defense final of the Italian competition “CyberChallenge.IT” [39].

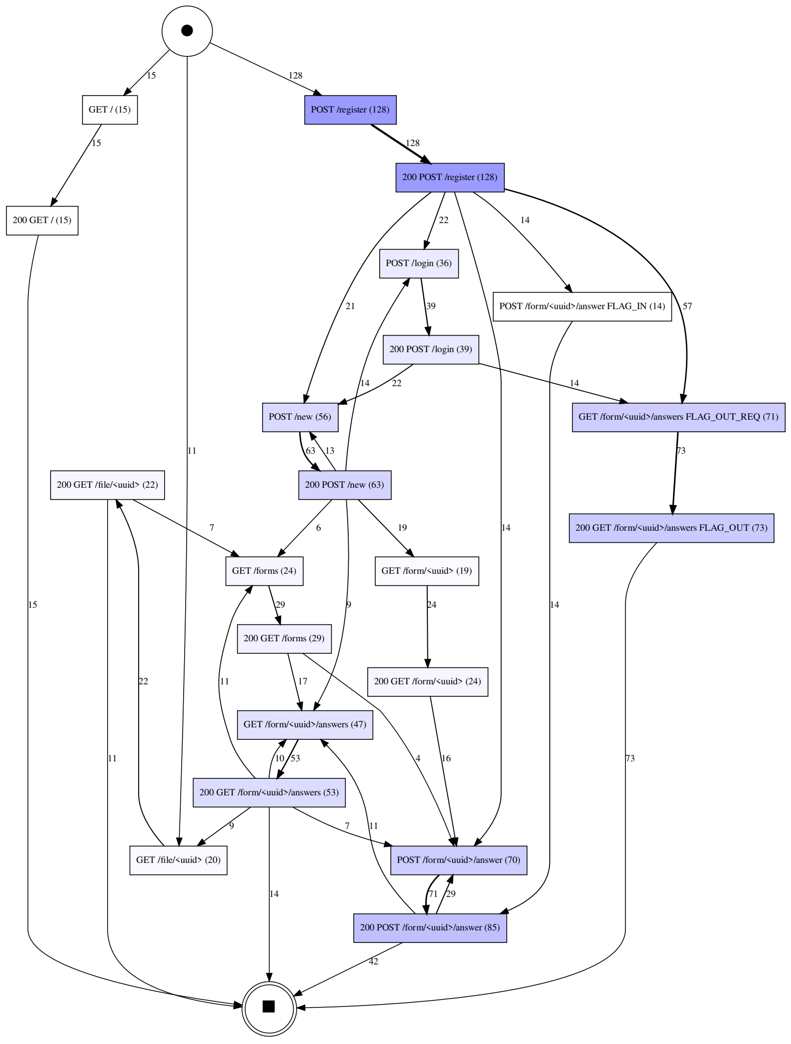
GET / (15)
POST /register (128)
200 POST /register (128)
200 GET / (15)
POST /login (36)
POST /form/<uuid>/answer FLAG_IN (14)
200 POST /login (39)
POST /new (56)
GET /form/<uuid>/answers FLAG_OUT_REQ (71)
200 GET /file/<uuid> (22)
200 POST /new (63)
200 GET /form/<uuid>/answers FLAG_OUT (73)
GET /forms (24)
GET /form/<uuid> (19)
200 GET /forms (29)
200 GET /form/<uuid> (24)
GET /form/<uuid>/answers (47)
200 GET /form/<uuid>/answers (53)
GET /file/<uuid> (20)
POST /form/<uuid>/answer (70)
200 POST /form/<uuid>/answer (85)

Figure 3: CCForms Challenge - Checkers and attack graph.

4.3.1. Evaluation

In order to properly evaluate the proposed approach, a set of experiments is carried out in different scenarios, and the resulting network logs are collected and processed. In the first scenario, the services from the vulnbox are deployed and the game server is running the checker script simulating lawful behavior. An accurate visualization of the checkers work is shown in Figure 2. In the second scenario, an attacker that actively sends payloads to the vulnbox is added, simulating a combined lawful and malicious behavior (see Figure 3). Through a “visual difference” between the previously described scenarios, it is possible to infer the interaction of the attackers with the system (i.e. the path that connects the “/register” node to the “/form/<uuid>/answers” response node marked as “FLAG_OUT”).

To support our claim, we ran a scenario comprising only the attacker while omitting the checker. This resulted in the graph shown in Figure 4, which corresponds to the previously identified path. This type of validation was possible because the experiments were conducted in a controlled environment.

A network log of 10 MB has been recorded for each simulated scenario.

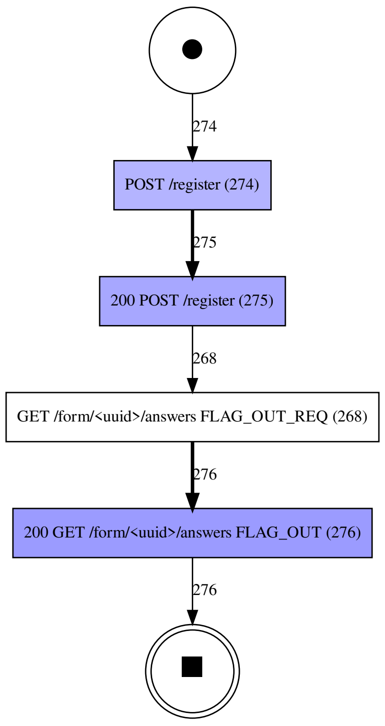
274
POST /register (274)
275
200 POST /register (275)
268
GET /form/<uuid>/answers FLAG_OUT_REQ (268)
276
200 GET /form/<uuid>/answers FLAG_OUT (276)
276

Figure 4: CCForms Challenge - Attack graph.

5. Conclusion and Future Work

This work presented an innovative approach that employs process mining techniques to extract attack graphs, in the form of Directly-Follows Graphs, from network logs. The effectiveness of the developed technique has been evaluated by applying it to network logs collected in a Cyber Range environment specifically tailored for this experimentation. The obtained DFGs proved to be accurate and effective in illustrating the primary attack paths and in helping to distinguish between malicious actions and legitimate user requests. These behavioral models are valuable tools for: understanding attackers' behavior; identifying exploited vulnerabilities; gather useful insights for developing both defensive and offensive strategies in Attack and Defense (AD) Capture the Flag (CTF) scenarios. Future work is geared at: evaluating the approach with different types of attacks (other than HTTP-based) and vulnerabilities (other than IDOR); refine and expand the methodology to enhance its robustness and minimize the need for manual setup.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] C.-W. Ten, G. Manimaran, C.-C. Liu, Cybersecurity for critical infrastructures: Attack and defense modeling, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 40 (2010) 853–865. doi:10.1109/TSMCA.2010.2048028.

[2] L. Coventry, D. Branley, Cybersecurity in healthcare: A narrative review of trends, threats and

ways forward, Maturitas 113 (2018) 48–52. doi:https://doi.org/10.1016/j.maturitas.2018.04.008.

[3] H. Al-Mohannadi, Q. Mirza, A. Namanya, I. Awan, A. Cullen, J. Disso, Cyber-attack modeling analysis techniques: An overview, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2016, pp. 69–76. doi:10.1109/W-FiCloud.2016.29.

[4] P. K. Manadhata, J. M. Wing, An attack surface metric, IEEE Transactions on Software Engineering 37 (2011) 371–386. doi:10.1109/TSE.2010.60.

[5] E. Hutchins, M. Cloppert, R. Amin, Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, Leading Issues in Information Warfare & Security Research 1 (2011).

[6] X. Lin, P. Zavarsky, R. Ruhl, D. Lindskog, Threat modeling for csrf attacks, 2009, pp. 486 – 491. doi:10.1109/CSE.2009.372.

[7] S. Caltagirone, A. D. Pendergast, C. Betz, The diamond model of intrusion analysis, 2013. URL: https://api.semanticscholar.org/CorpusID:108270876.

[8] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, E. Weippl, Dark clouds on the horizon: Using cloud storage as attack vector and online slack space, in: 20th USENIX Security Symposium (USENIX Security 11), 2011.

[9] C. Phillips, L. P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 Workshop on New Security Paradigms, NSPW '98, Association for Computing Machinery, New York, NY, USA, 1998, p. 71–79. URL: https://doi.org/10.1145/310889.310919. doi:10.1145/310889.310919.

[10] B. Schneier, Attack trees, Dr. Dobb's journal 24 (1999) 21–29.

[11] W. van der Aalst, Data Science in Action, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp. 3–23. URL: https://doi.org/10.1007/978-3-662-49851-4_1. doi:10.1007/978-3-662-49851-4_1.

[12] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, et al., Process mining manifesto, in: Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9, Springer, 2012, pp. 169–194.

[13] M. Macak, L. Daubner, M. F. Sani, B. Buhnova, Process mining usage in cybersecurity and software reliability analysis: A systematic literature review, Array 13 (2022) 100120.

[14] Y. Bertrand, B. Van den Abbeele, S. Veneruso, F. Leotta, M. Mecella, E. Serral, A survey on the application of process discovery techniques to smart spaces data, Engineering Applications of Artificial Intelligence 126 (2023) 106748. URL: https://www.sciencedirect.com/science/article/pii/S0952197623009326. doi:https://doi.org/10.1016/j.engappai.2023.106748.

[15] H. R'bigui, C. Cho, Heuristic rule-based process discovery approach from events data, International Journal of Technology Policy and Management 19 (2018). doi:10.1504/IJTPM.2019.10025752.

[16] W. M. Van der Aalst, B. F. van Dongen, C. W. Günther, A. Rozinat, H. Verbeek, A. Weijters, Prom: The process mining toolkit, in: Proceedings of the BPM 2009 Demonstration Track (BPMDemos 2009, Ulm, Germany, September 8, 2009), CEUR-WS. org, 2009, pp. 1–4.

[17] W. M. van der Aalst, A practitioner's guide to process mining: Limitations of the directly-follows graph, Procedia Computer Science 164 (2019) 321–328. doi:https://doi.org/10.1016/j.procs.2019.12.189, cENTERIS 2019 - International Conference on ENTERprise Information Systems / ProjMAN 2019 - International Conference on Project MANagement / HCist 2019 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2019.

[18] M. A. Siddiqui, J. W. Stokes, C. Seifert, E. Argyle, R. McCann, J. Neil, J. Carroll, Detecting cyber attacks using anomaly detection with explanations and expert feedback, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 2872–2876. doi:10.1109/ICASSP.2019.8683212.

[19] M. Alabadi, Y. Celik, Anomaly detection for cyber-security based on convolution neural network : A survey, in: proc of 2021 6th International Symposium on Computer and Information Processing Technology (ISCIPT), 2020, pp. 1–14. doi:10.1109/HORA49412.2020.9152899.

[20] F. Blefari, F. A. Pironti, A. Furfaro, Toward a log-based anomaly detection system for cyber range platforms, in: Proceedings of the 19th International Conference on Availability, Reliability and Security, ARES 24, Association for Computing Machinery, New York, NY, USA, 2024. URL: https://doi.org/10.1145/3664476.3669976. doi:10.1145/3664476.3669976.

[21] S. He, J. Zhu, P. He, M. R. Lyu, Experience report: System log analysis for anomaly detection, in: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE), IEEE, 2016, pp. 207–218.

[22] B. Hallaq, A. Nickolson, R. Smith, L. Maglaras, H. Janicke, K. Jones, CYRAN: A hybrid cyber range for testing security on ICS/SCADA systems, 2016, p. 16. doi:10.4018/978-1-5225-1829-7.ch012.

[23] NIST, The cyber range guide, 2024. URL: https://www.nist.gov/system/files/documents/2023/09/29/The%20Cyber%20Range_A%20Guide.pdf.

[24] G. Hatzivasilis, S. Ioannidis, M. Smyrlis, G. Spanoudakis, F. Frati, C. Braghin, E. Damiani, H. Koshutanski, G. Tsakirakis, T. Hildebrandt, L. Goeke, S. Pape, O. Blinder, M. Vinov, G. Leftheriotis, M. Kunc, F. Oikonomou, G. Magilo, V. Petrarolo, A. Chieti, R. Bordianu, The threat-arrest cyber range platform, in: 2021 IEEE International Conference on Cyber Security and Resilience (CSR), 2021, pp. 422–427. doi:10.1109/CSR51186.2021.9527963.

[25] T. Debatty, W. Mees, Building a cyber range for training cyberdefense situation awareness, in: 2019 International Conference on Military Communications and Information Systems (ICMCIS), 2019, pp. 1–6. doi:10.1109/ICMCIS.2019.8842802.

[26] P. Martou, K. Mens, B. Duhoux, A. Legay, Generating virtual scenarios for cyber ranges from feature-based context-oriented models: A case study, COP '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 35–43. doi:10.1145/3570353.3570358.

[27] V. Orbinato, A next-generation platform for cyber range-as-a-service, 2021. arXiv:2112.11233.

[28] G. Costa, E. Russo, A. Armando, Automating the generation of cyber range virtual scenarios with vsdl, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 13 (2022) 61–80. doi:10.58346/JOWUA.2022.I4.004.

[29] E. Ukwandu, M. A. B. Farah, H. Hindy, D. Brosset, D. Kavallieros, R. Atkinson, C. Tachtatzis, M. Bures, I. Andonovic, X. Bellekens, A review of cyber-ranges and test-beds: Current and future trends, Sensors 20 (2020). doi:10.3390/s20247148.

[30] C. Pham, D. Tang, K.-i. Chinen, R. Beuran, Cyris: A cyber range instantiation system for facilitating security training, in: Proceedings of the 7th Symposium on Information and Communication Technology, 2016, pp. 251–258.

[31] A. Furfaro, A. Piccolo, A. Parise, L. Argento, D. Saccà, A cloud-based platform for the emulation of complex cybersecurity scenarios, Future Generation Computer Systems 89 (2018) 791–803. doi:https://doi.org/10.1016/j.future.2018.07.025.

[32] D. Mugisha, Cyber security: Improving cyber defense through coherent joint red team and blue team, The Journal of Defense Modeling & Simulation 1 (2019).

[33] H. Jiang, T. Choi, R. K. L. Ko, Pandora: A cyber range environment for the safe testing and deployment of autonomous cyber attack tools, in: S. M. Thampi, G. Wang, D. B. Rawat, R. Ko, C.-I. Fan (Eds.), Security in Computing and Communications, Springer Singapore, Singapore, 2021, pp. 1–20.

[34] The Tcpdump Group, tcpdump, The Tcpdump Group, 2024. URL: https://www.tcpdump.org/.

[35] P. S. S. GmbH, Proxmox - powerful open-source server solutions, 2024. URL: https://www.proxmox.com/en/.

[36] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, L. F. M. Vieira, Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications, ACM Comput. Surv. 53 (2020). URL: https://doi.org/10.1145/3371038. doi:10.1145/3371038.

[37] K. Kominis, Pyshark: A python wrapper for tshark, allowing packet parsing, https://github.com/KimiNewt/pyshark, 2024. Accessed: 2024-12-16.

[38] Wireshark Foundation, Wireshark and TShark, 2024. URL: https://www.wireshark.org/.

[39] Cyberchallenge.it/, Cyberchallenge.it, 2024. URL: https://cyberchallenge.it/.