From Behavior to Breach: Modeling, Simulating and Detecting Attacks with Process Mining

Luca D'Este, Emil Løvstrand Mortensen, Emil Pontoppidan Rasmussen, Carlos E. Budde, Nicola Dragoni, and Andrea Burattin

DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark
{lucde,cesbu,ndra,andbur}@dtu.dk

Abstract. As digital systems expand in scope and complexity, the sophistication and frequency of cyber attacks increase as well. Digital systems keep track of all relevant events in system logs, which can be used for further analysis. This paper investigates the use of process mining on system logs to identify whether cyber attacks can be found and, if this is the case, which attacks have happened (either known or unknown ones, i.e., zero-day). To accomplish this goal we propose to formally model benign and attack behaviors and use process mining and conformance checking for detection. Additionally, having formal models allows to generate synthetic data referring to both benign behavior and attacks. The paper demonstrates the viability of the presented techniques using a real-world case study that analyzes Windows Security Logs to identify cyber attacks, modeled from the MITRE ATT&CK framework.
The contribution of this paper comprises: the modeling of benign behavior and cyber attacks in a formal way, the generation of conformant attack logs, the conformance-based attacks detection, and a real-world case study where all the contributions are validated. All data, models, and code are available online.

Keywords: Process Mining · Intrusion Detection · Cybersecurity · Conformance checking.

1 Introduction

Modern computing infrastructures, including enterprise networks, cloud platforms, and industrial control systems, produce large volumes of event logs that record sequences of actions performed by users, applications, and system components. These logs constitute a primary source of information for cybersecurity monitoring, where Security Operations Centres analyze system activity to detect malicious behavior such as unauthorized access, lateral movement, privilege escalation, or data exfiltration [20].

As digital infrastructure scales, the surface area exposed to adversarial exploitation also increases. The efficacy of defense strategies is inherently linked to the visibility of the actions that have occurred. System monitoring and logging provide visibility by recording observable, timestamped events produced by

users and processes [14]. These logs capture what a system did, and in what sequence. In the event of a security incident, these logs serve as primary evidence of adversarial behavior, revealing what an attacker accessed, which tools were invoked, and how the execution unfolded [14].

In addition, attackers are increasingly exploiting legitimate system functionality or previously unknown vulnerabilities (so-called "zero-day") [43], therefore traditional signature-based detection techniques [10,9] are often insufficient. For this reason, many security systems rely on *anomaly-based intrusion detection* (AIDS), where attacks are identified as deviations from a model of the *intended* system behavior. One category of AIDS is the "knowledge-based" system, which requires a *knowledge base* to identify legitimate activities [17].

In this paper, we propose the construction of a knowledge-based intrusion detection system that leverages formal process models [29]. Specifically, a process model can capture the legitimate sequence of activities into a *benign model* and, by employing process mining techniques [3,8], we have the ability to simulate as well as monitor execution traces to precisely characterize deviations from benign behavior. In addition, process models allow us to explicitly model attacks as perturbations of the benign behavior.

The contributions of this paper are threefold:

C1: **Behavioral modeling.** We show that process models can be used to capture benign system behavior as well as known cyber attacks, resulting in a set of models that collectively represent both benign and malicious execution scenarios (Section 3.3).

C2: **Simulation of events, including attacks.** Synthetic traces can be generated via the simulation of the behavioral models to mimic both benign and attack logs, thus eliminating the need for manual attack execution on sandboxes or data harvesting, which can be a complex, tedious, and incomplete task (Section 3.4).

C3: **Conformance-based detection.** Given an execution trace, we present the idea of evaluating it against all available models via conformance checking. The resulting scores are used to classify the trace as benign, attributable to a known attack type, or as a potential zero-day attack (Section 3.5).

All these contributions, from the modeling of normal behavior and the specification of attacks to simulation and attack detection, are showcased in a real-world case study based on the Windows Logon Process.[1]

The rest of the paper is structured as follows: Sect. 2 presents the relevant works, gaps in the literature, and the research questions. Sect. 3 describes the overall methodology and the primary contribution. Sect. 4 provides a thorough validation of the methodology. Finally, Sect. 5 concludes the paper. All data, models, and code are available online.[2]

[1] See `https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/windows-logon-scenarios`.

[2] See `https://doi.org/10.5281/zenodo.21835477`.

2 Background and Related Work

A prerequisite to an anomaly detection system is the characterization of the benign behavior. Process model discovery [4] has been identified as a solution to this challenge by generating models that are human-readable and formally grounded, such as Petri Nets [29], BPMN diagrams [15], or directly-follows graphs [3]. These models express the control-flow logic underlying benign behaviors. Previous attempts [33,34] to apply process mining to understand attack behavior focus on modeling the attacks, rather than on detecting them, and the corresponding analyses were conducted at a different level of granularity than what we are interested in this work.

Simulation represents a key pillar in the practice of cybersecurity [16]. Malware sandboxes are software environments that execute potentially malicious code in an isolated and instrumented manner. These sandboxes are then analyzed, and the resulting behavioral traces are used to identify and characterize threats [12]. Beyond the realm of malware analysis, the concept of simulation plays a pivotal role in red-team operations and penetration testing [39]. In these contexts, security practitioners meticulously emulate adversarial Tactics, Techniques, and Procedures against target environments, thereby facilitating the controlled examination of system vulnerabilities [26]. In the process mining literature, several studies have developed models that are subjected to manual perturbation for the purpose of cyber threat modeling, with their resulting behavior subsequently simulated [19,25]. Huang et al. take a different approach, introducing a framework that generates synthetic logs by defining attacks through the MITRE ATT&CK taxonomy [42] and injecting them into normal logs [13]. They also train a detection model on these logs for classification purposes. However, their architecture is dependent on existing logs, rather than being grounded on a formal process model.

The detection of deviations at the process level is facilitated by the discovery (either manual or automatic or both) of a benign process model from event logs of benign activity, followed by the application of conformance checking techniques, like *Token-based Replay Fitness* [35] and *ETC Precision* [28], to new event logs. This approach enables the identification of anomalous sequences of activities, not only of anomalous individual events. Myers et al. propose a detection framework for Industrial Control Systems for process-level attacks [30]. Their architecture employs process discovery to obtain a benign model and conformance checking to detect attacks. Nevertheless, the result of the experiment is limited to a binary outcome. Samiri et al. employ process mining to construct a benign model and a log of a man-in-the-middle attack [37]. This attack is executed in a secure environment and then compared to the model using conformance checking for the purpose of detection. The study further emphasizes the explainability and interpretability of the different behaviors. Nevertheless, their emphasis on modeling the attack is limited. The closest work is from Blefari et al. where process mining is used to detect anomalies and attack models are built [7]. However, the injection of attacks is limited and the resulting attack models are significantly different from the normal behavior.

The existing literature establishes three individually mature threads: *(i)* Process discovery techniques that accurately characterize normal and known attack behavior, *(ii)* Simulation that generates realistic attack traces for model construction and evaluation, *(iii)* Conformance checking methods that detect deviations from normative models. What remains largely unexplored is the integration of these into a coherent detection architecture for intrusion detection. The present work aims to bridge this gap and shows how the complete pipeline can be configured using process mining and leveraging formal process models. The approach is presented by showing attacks from the MITRE ATT&CK framework, and conformance checking alignments are leveraged as the detection primitive. The result is a system that is both generalizable to novel attack variants and interpretable for analyst review.

Therefore, from the current state of the art, we identified the following research questions:

RQ1: How can benign behavior be formally modeled and how can known attacks be expressed in such models in order to create process models of these attacks?
RQ2: How can traces from these models be simulated in a way that reflects both benign and malicious behavior, and how can the fidelity of the simulated traces with respect to real-world logs be verified?
RQ3: How can the benign and attack process models, in conjunction with conformance checking techniques, be used to classify traces as benign, known attack, or as a potential zero-day attack?

3 An Intrusion Detection Methodology

This section presents an overview of the complete methodology, defines the terminology needed, and explains the contributions of this paper.

3.1 Overview of the Contribution

The overall contribution of this paper is explained in Fig. 1. The typical behavior of a real system is captured in a model of the system that explicitly targets the benign behavior. Process modeling languages can be used for this task, such as Petri Nets [29] or DECLARE [32]. Such a model can be automatically inferred [3] or manually refined [38]. Since process models are used, an expert can adjust the underlying behavior to include the presence of a known attack; with this, we model how a system is expected to behave when a given attack is in progress. By repeating this procedure for a catalog of attacks [31], we end up with a complete library of processes, each modeling how the system is expected to behave when the corresponding attack is ongoing, i.e., threat modeling [45].

The second contribution consists of a simulation component. The idea is to consider any model produced during the modeling phase and generate a sequence of events that reflects it. For this specific task, an existing process simulator [18]

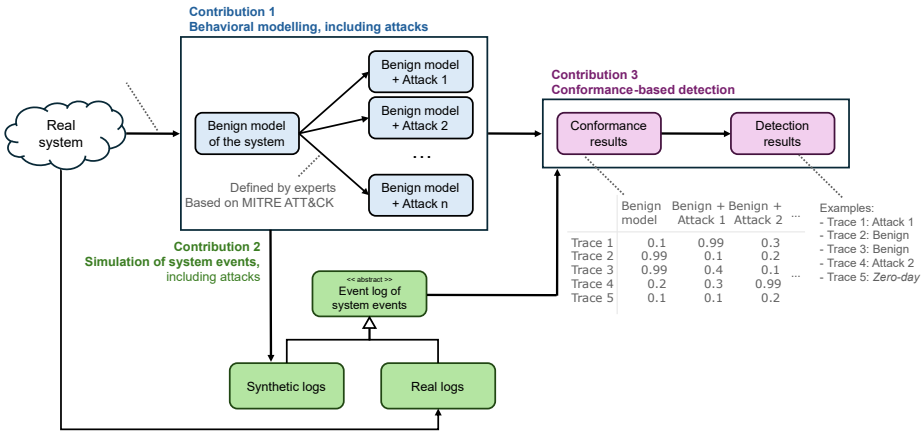

Fig. 1. Conceptual components of the paper, with the 3 contributions.

is used. The relevant aspect, in this case, is that it is now possible to obtain a log of events where attacks occurred, thus mitigating the problems of scarcity and imbalance of attack data [21,20].

The final contribution is the main intrusion detection component. This component has two steps: the first computes the conformance of each trace against each of the defined models (i.e., the benign model and all the models including attacks); the second step is about deciding, for each trace and based on the conformance results, how to classify the trace as either benign, or containing an known attack or none of the above, hence a so called "zero-day attack" [6].

In the rest of this section, first some basic terminology is introduced, and then each component is described in detail.

3.2 Basic Terminology

This section presents the basic definitions needed to delineate the fundamental concepts of the paper. Each definition is accompanied by a running example.

Definition 1 (Process [11]) *We reuse a definition of process from the literature [11]: "a collection of inter-related events, activities, and decision points that involve a number of actors and objects, which collectively lead to an outcome that is of value".*

In the context of cybersecurity, an example of a process is user authentication in an operating system. This process captures a common interaction pattern in which a user attempts to authenticate by inserting the password, and then either succeeds or fails to authenticate, therefore having access granted or denied.

While processes are just conceptual descriptions of how certain goals are achieved, for our purposes, we need to materialize them into formal models that

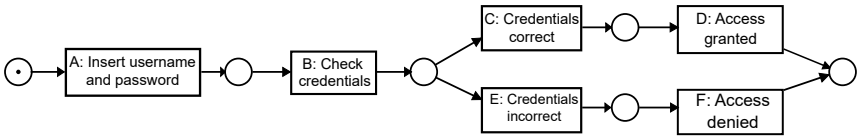

Fig. 2. Process model for the simplified authentication process.

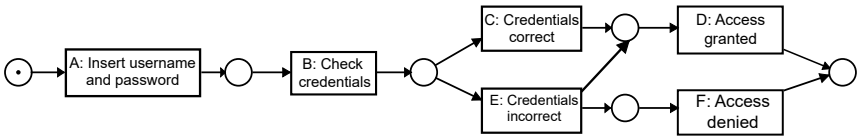

Fig. 3. Process model of the simplified authentication process with an attack.

can be used for further computation. In this paper, however, we do not restrict the specification to any concrete modeling language.

Definition 2 (Process Model) *A model is a finite, compact representation of the intended behavior of a process, defining which activities can be performed and in which order. A model characterizes the set of all permitted execution sequences, which may be infinite. Formally, a model* $M \in \mathcal{M}$ *from the set of all possible models over the set of activities* A *induces a language* $\mathcal{L}(M) \subseteq A^*$*, where* A^* *denotes the set of all finite sequences over* A*. Each sequence* $\sigma \in \mathcal{L}(M)$ *represents a valid execution of the process according to* M*.*

Concrete examples satisfying this definition are Petri Nets [29] and DECLARE [32]. Fig. 2 displays a Petri Net capturing a simplified version of the user authentication process. This model shows that the first activity is inserting the username and the password, and then the system verifies whether these are correct or not. After that, access is granted or denied based on the previous activity. The model therefore shows which activities have to happen and in which order. Please note that in the rest of the paper we will refer to these activities as A, B, C, D, E, and F (as reported in the figure).

Since we have an explicit model of our benign authentication process, we can decide to model how a given attack would be expressed. Fig. 3 shows such a model where, after credentials are verified as incorrect, access can still be granted.

Definition 3 (Event, Trace, Log) *An event is the recording of the execution of an activity as part of an instance of a process. A trace is the sequence of events that belong to the same process instance. A log is a collection of traces. More formally, a log is defined as* $L = \{t_1, t_2, \ldots, t_n\}$ *where each trace* $t_i =$

$\langle a_1, a_2, \ldots, a_k \rangle \in A^*$ *is a finite sequence of events, where each event* $a_j \in A$ *is the activity name that has happened.*[3]

Considering the process model from Fig. 2, a single trace may record a failed attempt to logon, preceded by an attempt to insert the password: $\langle A, B, E, F \rangle$. A log can contain multiple repetitions of the same trace. For instance:

$$\left\{ \langle A, B, C, D \rangle^{13}, \langle A, B, E, F \rangle^{17} \right\}.$$

Definition 4 (Simulation) *Simulation is the process of generating a complete trace from a model. Formally, we define a simulation as a function* $Simulation : \mathcal{M} \to A^*$. *Repeating the same procedure multiple times yields a log.*[4]

When applied to the running example from Fig. 2, a simulation may produce traces such as: $\langle A, B, C, D \rangle$ reflecting a scenario in which a successful authentication happens, or $\langle A, B, E, F \rangle$ for a failed authentication. The model containing the attack from Fig. 3 can also simulate $\langle A, B, E, D \rangle$, emulating the successful attack.

Definition 5 (Conformance checking) *Conformance checking is a function that measures the degree to which a trace adheres to a reference process model by comparing the given trace to the closest trace in the model language. Formally, conformance checking is defined as a function* $Conformance : A^* \times \mathcal{M} \to [0, 1]$, *where a value of* 1 *denotes perfect adherence (i.e, the trace belongs to the language of the reference process model language* $\mathcal{L}(M)$*) and lower values reflect the distance of the trace to the most similar one in the reference model language.*

Considering the example Fig. 2, the conformance on trace $\langle A, B, C, D \rangle$ would be assigned a score of 1. Conversely, trace $\langle B, A, C, D \rangle$ would get a lower score.

When several models of behavior are available and a trace is provided, conformance checking can be used to determine which model best matches it. We call this detection:

Definition 6 (Detection) *The process of attributing a given trace to one of a set of reference models. This attribution is based on the degree to which the observed trace conforms to each reference model. Formally we can define* $Detection : \mathcal{P}(\mathcal{M}) \times A^* \to \mathcal{M}$, *where* $\mathcal{P}$ *denotes the power set.*

[3] We are simplifying the definitions [3] of event, trace, and log to focus only on what is relevant to us. In particular, in our event and trace definitions, we do not consider anything but the activities executed and their order.

[4] We simplify the definition of simulation, since the way it resolves the non-determinism of the process model (e.g., by following probability distributions) is not relevant to this work.

3.3 Modeling Benign Behavior and Attacks

This section analyzes contribution C1 (see Fig. 1) to answer research question RQ1. In particular, it focuses on the modeling phase to produce a set of process models that will be used to compare traces at a later stage. As part of this phase, we need to discuss both modeling the benign process model as well as modeling the benign model when an attack is injected.

The benign model represents all the possible sequences of events that are expected during normal system operation. This model can be automatically discovered using a control-flow discovery algorithm [4] and it can also be manually revised and certified. To automatically discover the benign model, data has to be collected during normal system operation. It is important to ground the model in empirical data to ensure that the model reflects the process as it actually unfolds in practice. This is particularly relevant, since the type of attacks we aim to identify can consist only of a scrambled order of the activities already present in the model.

Once a benign model is available, it is possible to start modeling how attacks affect it. Specifically, we propose creating a new model for each attack we consider. Each attack model consists of a perturbation of the activities in the benign model: new activities can be inserted, activities can be removed, or the order of existing activities can be altered. Such attack modeling (a.k.a. threat modeling) requires an expert to deeply understand the underlying system as well as the attack they aim to model and combine them. In the cybersecurity industry, there are several libraries of possible attacks, such as MITRE ATT&CK.[5] When an attack is injected into a benign model, it should be adjusted to include only those activities relevant to the observed system from the benign model's perspective, i.e., the model represents the monitored asset from a *defensive* standpoint. Additionally, the injected behavior encodes the attack in such a way that every execution of the attack model is guaranteed to contain the attack behavior.

Different systems can be modeled by applying the same pipeline (i.e., benign model identification followed by attacks modeling) to a different set of data.

3.4 Simulation of Benign Behavior and Attacks

In this section we focus on contribution C2 (see Fig. 1) to answer RQ2. In cybersecurity, it is very challenging to collect data containing attacks [21]. Oftentimes, only limited attacks are available, thus creating very imbalanced datasets [20]. A possible way around these problems is using honeypots or sandboxes [27] to simulate attacks in a controlled environment. Some attacks, however, might be very challenging to replicate or simulate [24].

The construction of models described in the previous section serves as the foundation to simulate realistic behavior. These models can be used to generate synthetic logs that mirror both benign and each of the modeled attack behaviors, thereby producing a labeled collection of traces that encompasses the full

[5] See `https://attack.mitre.org/`.

Algorithm 1: Conformance-based Intrusion Detection

Input: t: trace to be classified
M_b: process model expressing the benign behavior
$M_{A_1}, \ldots, M_{A_n}$: process models expressing attacks $A_1, \ldots, A_n$
τ: threshold for minimum conformance
Output: Classification of the trace as benign, known attack, or zero-day

$M \leftarrow \{M_b\} \cup \{M_{A_1}, \ldots, M_{A_n}\}$
▷ Extract the model that has the highest conformance w.r.t. the trace
$M_{target} \leftarrow \underset{m \in M}{\text{argmax}}\ \mathit{Conformance}(t, m)$
if $\mathit{Conformance}(t, M_{target}) > \tau$ **then** ▷ If the conformance is above the minimum threshold, then extract the corresponding classification
  if M_{target} *is* M_b **then** ▷ The model with the highest conformance is the benign one, so the trace is also benign
    return trace classified as benign
  else ▷ The trace belongs to an attack model, returns corresponding attack
    return trace classified as attack M_{target} ▷ $M_{target} \in \{M_{A_1}, \ldots, M_{A_n}\}$
  end
else ▷ No model has high enough conformance, so it possibly is a zero-day attack
  return trace classified as potential zero-day attack
end

range of observations. The simulation of a process model is a complicated topic in itself, with several parameters that require configuration [2]; however, since our attack models are required to guarantee (cf. previous section) that every execution contains the attack behavior, each execution (regardless of the configured simulation parameters) will express this trait.

3.5 Conformance-based Intrusion Detection

This section analyzes contribution C3 (see Fig. 1) to answer RQ3. The fundamental idea of conformance-based intrusion detection lies in the fact that, given an execution trace and given a family of process models, one expressing the benign behavior and others expressing an attack behavior, we can use conformance checking to measure whether the given trace expresses benign behavior, a known attack, or neither.

The classification of a given trace is determined by computing the conformance of the trace against the benign model as well as all attack models. As presented in Algorithm 1, first, the conformance is computed against all models, and the one with the highest score is identified. If the highest conformance is above a given threshold, then the trace is classified either as a benign one (in case the top score is achieved against the benign model) or as an attack trace (in case the top score is achieved against an attack trace). If no model produces a score greater than the threshold, then it means that the trace does not belong to the benign model, nor to any of the known attacks, so it is a potential zero-

day attack. It is important to note that our results are more extensive than the output of conventional anomaly detection methods: rather than a binary outcome (i.e., benign/anomalous), our approach allows to distinguish which attack is going on or even whether an unknown attack is happening. The final case is particularly valuable in the cybersecurity domain, as it extends the detection capability beyond the set of threats modeled.

The current approach assumes traces to capture complete execution and thus, from a cybersecurity point of view, it is mostly suitable as a forensics [14] technique.

4 Case Study: Intrusion Detection on the Windows Logon Process

The case study presented in this section offers a comprehensive illustration of the previously outlined methodology. This methodology is employed to identify and detect attacks on the Windows Logon Process, which are recorded in the Windows Security Logs.[6] This source of log is an essential resource in real-world cyberdefense [41]. The significance of such a log is reinforced by its widespread availability, a factor that enhances the reproducibility of the pipeline. The case study is also generalizable, as it does not require any supplemental assumptions.

The intended logon process is modeled as a Petri Net. A selected number of attacks, presented in MITRE ATT&CK, are manually injected into such a model. These attacks are intended to demonstrate the versatility of the proposed technique in terms of attack strategies, as these represent three different types of "control-flow attacks":

1. The "Red Flag" attack corresponds to the *Modify Registry*, ATT&CK ID T1112.[7] It includes an activity that is sufficiently suspicious alone.
2. The "Repeat" attack is the *Password Guessing*, ATT&CK ID T1110.001.[8] It comprises actions that are normally accepted, but in a repetition pattern.
3. The "Composite" attack corresponds to the *OS Credential Dumping: LSASS Memory*, ATT&CK ID T1003.001.[9] It comprises sequences of benign actions that are scrambled in a new way to create the attack.

These models are then simulated to mimic different hostile environments. Simulations are validated against real executions, and, finally, a conformance-based detection system is used to correctly classify each of these traces into their corresponding model.

[6] See `https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor`.
[7] See `https://attack.mitre.org/techniques/T1112/`
[8] See `https://attack.mitre.org/techniques/T1110/001/`.
[9] See `https://attack.mitre.org/techniques/T1003/001/`.

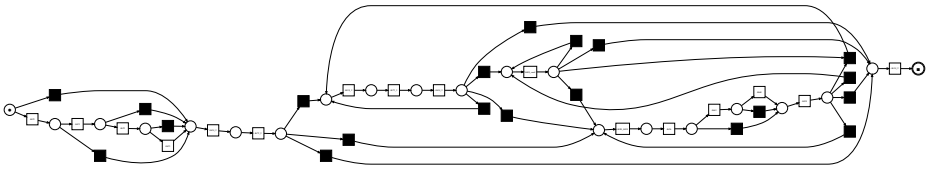

Fig. 4. Petri Net of the benign process capturing the Windows Logon Process. A high-resolution version of the picture is available at `https://github.com/processintelligence/ids/tree/EDOC2026/Models`.

4.1 Benign and Attacks Process Models

The first step consists of constructing the benign model of the Windows Logon Process, based on the Windows Security Logs. A detailed description of how events are recorded in the Windows Security Logs and what they represent is available in the literature [44].

In order to construct events and generate logs, a virtual machine has been set up. On this machine, the target process has been executed several times to derive a log. The resulting log is both meaningful and high-fidelity, while concurrently preserving a degree of background noise, which is necessary to ensure the log complexity is accurate. The log generation has been automated to capture a broader range of behaviors by adopting an approach inspired by "smart generation-based fuzzing" [40]. It is relevant to mention that the benign process includes distinct logon types, following different process flows.

After constructing the log, multiple control-flow discovery algorithms [5,22,23] have been tested with varying parameters to derive a process model. Fitness [35] and precision [28] were used to identify the model achieving the optimal results. In the end, the Heuristics Miner (with dependency threshold: 0.5, AND-threshold: 0.9; loop-2 threshold: 0.9) led to the best results, with a fitness of 0.9875 and a precision of 0.8341. These high values for fitness and precision ensure that the benign model is a high-quality model, capable of replicating the observed behavior of the system while introducing limited additional behavior. Fig. 4 depicts the process model of the benign process.

With the benign model available, it is possible to inject this with the attack behavior. To accomplish this, an expert helped interpret the MITRE ATT&CK descriptions and adapt them to capture the right perspective. In particular, the most important aspect is guaranteeing that a complete run of the process is guaranteed to include the attack behavior.

The same procedure, i.e., including the attack behavior into the benign model, is repeated for all the attacks that we want to recognize. Fig. 5 shows the Petri Net with the benign model injected with the Repeat attack. This model requires at least five repetitions of attempting to authenticate, and so it accurately depicts the Password Guessing attack.

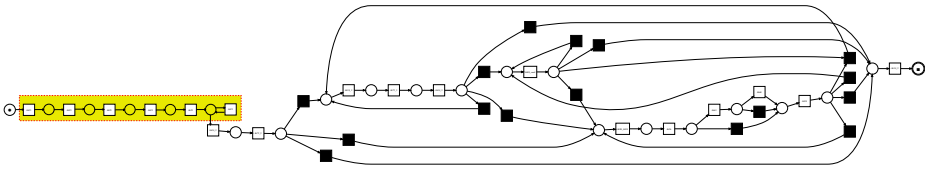

Fig. 5. Petri Net of the Windows Logon Process injected with the Repeat attack. The parts in yellow highlight the changes in behavior. A high-resolution version of the picture is available at `https://github.com/processintelligence/ids/tree/EDOC2026/Models`.

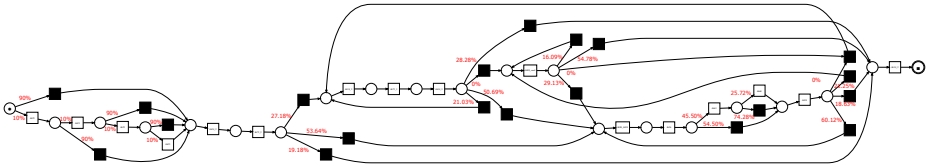

Fig. 6. Petri Net of the benign process capturing the Windows Logon Process augmented with probabilities at all points of nondeterminism. A high-resolution version of the picture is available at `https://github.com/processintelligence/ids/tree/EDOC2026/Models`.

4.2 Simulation of Benign Behavior and Attacks

Simulation allows obtaining traces expressing both the benign behavior as well as any attacks, without manually executing the attacks in the sandbox machine.

To ensure the authenticity of the simulation, two dimensions are considered: structural and frequential realism. *Structural realism* predicates that the set of traces is syntactically valid with respect to the underlying process. Existing generation methods such as heuristic simulations and large language model-based generators often exhibit deficiencies in structural coherence or controllability [36]. This results in synthetic data that does not accurately represent real-world system interactions. The enforcement of structural validity through a formal model, such as a Petri Net, ensures that only valid traces are generated. In the absence of a structural enforcement, a simulator may generate traces that are statistically plausible but process-impossible. Consequently, a detection model built on such traces would highlight spurious patterns that do not transfer to real data.

Conversely, the concept of *frequency realism* assesses the distribution of behavior over the generated traces. This involves determining which event sequences are observed and the rates at which they occur relative to one another. A simulator that is structurally realistic will generate traces uniformly over the language of the process. Such behavior, however, does not necessarily reflect what happens in real processes. In practice, oftentimes a small number of trace variants account for the majority of the log, while rare paths are executed only under exceptional conditions. For the simulation in our use case, we used

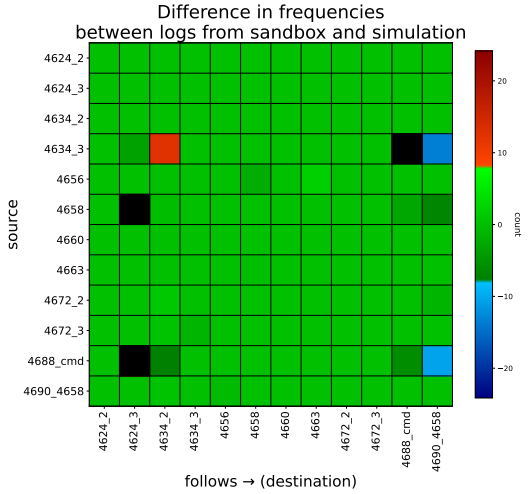

Fig. 7. Heat map representing the difference in frequencies of directly follow relations between the log extracted from the sandbox machine (ground truth) and the simulated one.

LogPPL [18][10]. Specifically, the benign model was transformed into a probabilistic program, thereby enabling the generation of event logs with statistical guarantees via probabilistic program executions. This process harnesses the simulation and inference engines of WebPPL[11] to impose distributional constraints directly on the simulation.

Transforming the model into one that can be converted into a probabilistic program requires enhancing the Petri Net with probabilities at all points of nondeterminism. A probability distribution was extracted from the log by using token replay and counting how many executions traversed each choice (i.e., point of nondeterminism). The Petri Net of the benign model, augmented with probabilities, is presented in Fig. 6. In the case of the models describing attacks, the probabilities were manually added by experts.

The simulation is entirely delegated to the LogPPL tool, which uses the assigned probabilities to control the choices allowed by the model. The probabilistic programming framework employs a sampling method from these probability distributions at runtime and fires the corresponding transition based on the outcome of each sample. This mechanism ensures that the simulation faithfully reflects the stochastic nature of the modeled process. As a result, execution traces are both statistically consistent with the defined probability assignments and representative of the behavioral variability inherent in the system.

[10] See `https://github.com/martinkuhn94/LogPPL`.
[11] See `http://webppl.org/`.

Table 1. Conformance values and classification (according to Alg. 1, considering $\tau = 0.9$) of each trace for the given benign and attack models.

	Benign trace	RedFlag trace	Repeat trace	Composite trace
Benign model	**1.000**	0.846	0.625	0.643
Model with RedFlag attack	0.962	**0.944**	0.476	0.667
Model with Repeat attack	0.808	0.611	**1.000**	0.545
Model with Composite attack	0.771	0.667	0.333	**0.976**
Classification	benign	attack RedFlag	attack Repeat	attack Composite

In order to validate the simulated logs, a *log-to-log* metric is used to compare the frequency of the directly-follows relations between the log extracted from the sandbox machine (i.e., our ground truth) and the simulated one. A heatmap showing the difference in frequencies of the directly-follows relations is presented in Fig. 7. Green squares represent relations with negligible differences in frequency between our ground truth and the simulated data. The findings suggest that the model, with the provided probabilities, is capable of reproducing behavior with both structural and frequential realism. Structural realism is highlighted by the scarcity of black cells (i.e., those present in the real log but not permitted by the model), confirming that the discovered control-flow closely mirrors legitimate system execution. Frequential realism is reflected in the limited number of blue and red cells, indicating those transitions whose relative frequency diverges between the simulated and real logs, confirming that the probability weights assigned to branching choices successfully reproduce the distributional characteristics of observed behavior, and not merely its structural skeleton.

4.3 Attack detection

For this case study, four distinct traces have been considered: one benign trace, one presenting the Red Flag attack, one with the Repeat attack, and one with the Composite attack. For each trace, the Intrusion Detection approach from Algorithm 1 is used, with an alignment-based *Conformance* function [1]. The results are presented in Table 1. The columns represent the traces, and the rows represent the models. It is interesting to observe that, considering the procedure in Alg. 1 with $\tau = 0.9$, all traces (both the benign and the attack) are correctly classified.

5 Conclusion and future work

This work introduces an approach for using process mining to model, simulate, and detect cyber attacks. This method allows, given a trace and a population

of models, the classification of the trace as being benign, belonging to a known attack, or a zero-day attack. The approach illustrates how process mining techniques can be meaningfully applied in a new domain: cybersecurity. A real-world case study in this domain, focusing on the Windows Logon Process, has been presented, demonstrating the technique's viability in a real-world context.

Considering the research questions that were previously introduced, RQ1 is addressed by C1 (Sect. 3.3), wherein it has been demonstrated how the benign model can be discovered from a common usage log source, such as the Windows Security Logs, and modeled using Petri Nets. Furthermore, C1 demonstrates the ability to model and inject attacks into the benign model. RQ2 is addressed by C2 (Sect. 3.4), which validated the quality of the discovered model and the similarity between the simulated traces and the original logs. This was achieved through a precise log-to-log comparison of directly-follow relations. Finally, RQ3 is answered by C3 (Sect. 3.5), which demonstrates that the architecture can attribute each trace to the corresponding model by means of a conformance checking technique.

This paper serves as a foundation upon which more sophisticated detection capabilities can be built. Specifically, with the right next steps, this work has the potential to improve the detection of cyber attacks, exploiting a more complete behavioral characterization. Future work will address the automation of attack model generation to maintain the library of attacks as rich and up-to-date as possible, and the conversion of the detection procedure into a streaming one, thus enabling the application of this technique in real-time.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Int. Rev. Data Min. and Knowl. Disc. **2**(2), 182–192 (Mar 2012). https://doi.org/10.1002/widm.1045
2. van der Aalst, W.M.P.: Business Process Simulation Survival Guide. In: Handbook on Business Process Management 1, pp. 337–370. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-642-45100-3{_}15
3. van der Aalst, W.M.: Process Mining. Springer Berlin Heidelberg, second edn. (2016). https://doi.org/10.1007/978-3-662-49851-4
4. van der Aalst, W.M.: Foundations of process discovery. In: Process Mining Handbook, pp. 37–75. Springer (2022)
5. van der Aalst, W.M., Weijters, T.A.J.M.M., de Medeiros, A.K.A.: Process Mining with the Heuristics Miner-algorithm. BETA Working Paper Series, WP 166 (2006)
6. Bilge, L., Dumitraş, T.: Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. p. 833–844. CCS '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2382196.2382284

7. Blefari, F., Pironti, F.A., Furfaro, A.: 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). https://doi.org/10.1145/3664476.3669976
8. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-99414-7
9. Chandola, V., Banerjee, A., Kumar, V.: Anomaly Detection: A Survey. ACM Computing Surveys **41**(3), 1–58 (7 2009). https://doi.org/10.1145/1541880.1541882
10. Denning, D.E.: An Intrusion-Detection Model. In: 1986 IEEE Symposium on Security and Privacy. pp. 118–118. IEEE (4 1986). https://doi.org/10.1109/SP.1986.10010
11. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, 2 edn. (2018)
12. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR) **44**(2), 1–42 (2008)
13. Huang, Y.T., Guo, Y.R., Yang, Y.S., Wong, G.W., Jheng, Y.Z., Sun, Y., Modini, J., Lynar, T., Chen, M.C.: Saga: Synthetic audit log generation for apt campaigns. Arxiv (cornell University) (2024). https://doi.org/10.48550/arXiv.2411.13138
14. Jakobsen, A.T.: Practical cyber intelligence : a hands-on guide to digital forensics. Wiley (2024)
15. Kalenkova, A., Burattin, A., de Leoni, M., van der Aalst, W.M., Sperduti, A.: Discovering high-level BPMN process models from event data. Business Process Management Journal **25**(5) (2019). https://doi.org/10.1108/BPMJ-02-2018-0051
16. Kavak, H., Padilla, J.J., Vernon-Bido, D., Diallo, S.Y., Gore, R., Shetty, S.: Simulation for cybersecurity: state of the art and future directions. Journal of Cybersecurity **7**(1), tyab005 (01 2021). https://doi.org/10.1093/cybsec/tyab005
17. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 20 (12 2019). https://doi.org/10.1186/s42400-019-0038-7
18. Kuhn, M., Grüger, J., Matheja, C., Rivkin, A.: Data petri nets meet probabilistic programming. In: Marrella, A., Resinas, M., Jans, M., Rosemann, M. (eds.) Business Process Management - 22nd International Conference, BPM 2024, Krakow, Poland, September 1-6, 2024, Proceedings. pp. 21–38. Lecture Notes in Computer Science, Springer (2024). https://doi.org/10.1007/978-3-031-70396-6_2
19. Kumar, R., Pandey, S., Das, D.: Palm: A framework to identify novel attacks in an e-commerce system. Proceedings of Ieee Pacific Rim International Symposium on Dependable Computing, Prdc pp. 143–152 (2024). https://doi.org/10.1109/PRDC63035.2024.00027
20. Landauer, M., Skopik, F., Frank, M., Hotwagner, W., Wurzenberger, M., Rauber, A.: Maintainable Log Datasets for Evaluation of Intrusion Detection Systems. IEEE Transactions on Dependable and Secure Computing **20**(4), 3466–3482 (7 2023). https://doi.org/10.1109/TDSC.2022.3201582
21. Larroche, C., Mazel, J., Clémençon, S.: Recent trends in statistical analysis of event logs for network-wide intrusion detection. In: Conference on Artificial Intelligence for Defense (CAID) (2020)

22. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.M., Desel, J. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 311–329. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
23. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) Business Process Management Workshops. pp. 66–78. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
24. Liu, S., Feng, P., Wang, S., Sun, K., Cao, J.: Enhancing malware analysis sandboxes with emulated user behavior. Computers & Security **115**, 102613 (4 2022). https://doi.org/10.1016/j.cose.2022.102613
25. Makwana, M.D., Thakkar, V., Das, D., Kumar, R.: Simulating cyber-attack scenarios by discovering petri-nets from large-scale event logs. In: 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS). pp. 49–54 (2024). https://doi.org/10.1109/COMSNETS59351.2024.10427052
26. Miller, D., Alford, R., Applebaum, A., Foster, H., Little, C., Strom, B.: Automated adversary emulation: A case for planning and acting with unknowns. Tech. rep., The MITRE Corporation (2018)
27. Morić, Z., Dakić, V., Regvart, D.: Advancing Cybersecurity with Honeypots and Deception Strategies. Informatics **12**(1), 14 (1 2025). https://doi.org/10.3390/informatics12010014
28. Munoz-Gama, J., Carmona, J.: A fresh look at Precision in Process Conformance. In: Proceedings of BPM. pp. 211–226. Springer Berlin / Heidelberg (2010). https://doi.org/10.1007/978-3-642-15618-2{_}16
29. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
30. Myers, D., Suriadi, S., Radke, K., Foo, E.: Anomaly detection for industrial control systems using process mining. Computers & Security **78**, 103–125 (2018). https://doi.org/10.1016/j.cose.2018.06.002
31. Nicoletti, S.M., Lopuhaä-Zwakenberg, M., Stoelinga, M., Massacci, F., Budde, C.E.: How Hard Can It Be? Quantifying MITRE Attack Campaigns with Attack Trees and cATM Logic. ACM Transactions on Software Engineering and Methodology (1 2026). https://doi.org/10.1145/3789665
32. Pešić, M., Schonenberg, H., van der Aalst, W.M.: DECLARE: Full Support for Loosely-Structured Processes. In: Proceedings of EDOC. pp. 287–298. IEEE (2007). https://doi.org/10.1109/EDOC.2007.14
33. Rodriguez, M., Betarte, G., Calegari, D.: A process mining-based approach for attacker profiling. 2021 Ieee Urucon, Urucon 2021 pp. 425–429 (2021). https://doi.org/10.1109/URUCON53396.2021.9647342
34. Rodríguez, M., Betarte, G., Calegari, D.: A process mining-based method for attacker profiling using the mitre att&ck taxonomy. Journal of Internet Services and Applications **15**(1), 212–232 (2024). https://doi.org/10.5753/jisa.2024.3902
35. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008). https://doi.org/10.1016/j.is.2007.07.001
36. Samanta, R.: Structurally valid log generation using fsm-gflownets. arXiv preprint arXiv:2510.26197 (2025)

37. Samiri, S.E., Lamghari, Z., Fakhar, R.: Towards the application of process mining for analyzing network security. Statistics, Optimization and Information Computing **15**(2), 1151–1161 (2026). https://doi.org/10.19139/soic-2310-5070-2557
38. Schuster, D., van Zelst, S.J., van der Aalst, W.M.: Cortado: A dedicated process mining tool for interactive process discovery. SoftwareX **22**, 101373 (5 2023). https://doi.org/10.1016/j.softx.2023.101373
39. Shmaryahu, D., Shani, G., Hoffmann, J., Steinmetz, M.: Simulated penetration testing as contingent planning. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 28, pp. 241–249 (2018)
40. Song, J., Alves-Foss, J.: A fuzzing tool based on automated grammar detection. Software **3**(4), 569–586 (2024). https://doi.org/10.3390/software3040028
41. Sreelekshmi, S., Gayathri, S., Gopika, S.: Unveiling windows security: Detecting security breaches using windows event logs. In: International Conference on Mathematical Modeling and Computational Science. pp. 372–384. Springer (2025)
42. Strom, B.E., Applebaum, A., Miller, D.P., Nickels, K.C., Pennington, A.G., Thomas, C.B.: MITRE ATT&CK: Design and philosophy. Tech. rep., The MITRE Corporation (2018)
43. Symantec: Internet Security Threat Report (ISTR). Tech. rep., Symantec (2017), https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf
44. Turcotte, M.J.M., Kent, A.D., Hash, C.: Unified Host and Network Data Set, chap. Chapter 1, pp. 1–22. World Scientific (nov 2018). https://doi.org/10.1142/9781786345646_001
45. Xiong, W., Lagerström, R.: Threat modeling – A systematic literature review. Computers & Security **84**, 53–69 (7 2019). https://doi.org/10.1016/j.cose.2019.03.010