Towards a Systematic Process-aware Behavioral Analysis for Security

Laura Genga and Nicola Zannone
Eindhoven University of Technology
{l.genga, n.zannone}@tue.nl

Keywords: Behavior Analysis, Process Mining.

Abstract: Nowadays, security is a key concern for organizations. An increasingly popular solution to enhance security in organizational settings is the adoption of anomaly detection systems. These systems raise an alert when an abnormal behavior is detected, upon which proper measures have to be taken. A well-known drawback of these solutions is that the underlying detection engine is a black box, i.e., the behavioral profiles used for detections are encoded in some mathematical model that is challenging to understand for human analysts or, in some cases, is not even accessible. Therefore, anomaly detection systems often fail in supporting analysts in understanding what is happening in the system and how to respond to detected security threats. In this work, we investigate the use of process analysis techniques to build behavioral models understandable by human analysts. We also delineate a systematic methodology for process-aware behaviors analysis and discuss the findings obtained by applying such a methodology to a real-world event log.

1 INTRODUCTION

The rapidly growing of security incidents and frauds during the last years, often due to insider threats and the lack of effective internal controls (Association of Certified Fraud Examiners, 2018; Richardson, 2008), has made security a key concern for organizations. Several research efforts have been devoted both by academics and practitioners to address such issues. An increasingly popular solution to enhance the security of an organization is represented by *anomaly detection* systems (Patcha and Park, 2007). These systems aim to analyze users' interactions with the system in order to detect abnormal behaviors and raise alerts when security threats are detected.

One of the main issues with these techniques is that they model system behaviors as a *black-box*. In particular, they build models of normal behavior that are not accessible or, anyway, not easily understandable by human analysts. This hampers the comprehension of what is happening in the system and, thus, makes it challenging to determine how to response to the raised alerts (Costante et al., 2016).

In this paper, we investigate the potentiality of data science, in particular of process analysis, techniques for deriving *white-box* models of normal system behavior, i.e. models that are actually accessible by a human analyst, to support the detection and analysis of anomalous behaviors.

The definition of accurate behavioral models for security is an undeveloped topic. Only a few work have investigated this issue and, to date, a systematic methodology is still missing. The main aim of this work is illustrating the benefits of applying process analysis techniques to explore security-related aspects, to stimulate future research in this direction. We also propose a methodology for a systematic behavior analysis for security. As a proof of concept, we exploit a real-world case study to show how existing process analysis techniques can be employed to build behavioral profiles tailored to the detection of security flaws within organizational settings. We discuss the main findings obtained from the event log at hand.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work on behavioral analysis for security. Section 3 delineates the proposed methodology and Section 4 discusses the main findings derived from a real-life case study. Finally, Section 5 draws conclusions and delineates directions for future work.

2 BEHAVIORAL ANALYSIS FOR SECURITY

Within the scope of this work, a *behavior* refers to a set of activities performed by one or more entities (either humans or software agents) on a given computational system, usually tracked by some logging mechanism (Cao, 2010). *Behavioral Analysis* aims to explore event logs to gain insights on the system behavior, e.g., typical behavioral patterns. In the security domain, behavioral analysis is often used to support *anomaly detection*. The goal of an anomaly detection system is to identify system behaviors that deviate from what it is considered to be the desired or normal behavior (Patcha and Park, 2007; Chandola et al., 2009; Costante et al., 2016).

Anomaly detection systems involve a *training* phase, where a *model* representing the normal system behavior is learned with respect to *features* of interests (e.g., CPU usage, combination of users' commands), and a *detection* phase, where the learned model is used to detect possible anomalies.

To build profiles of normal behavior, either supervised or unsupervised techniques are usually employed. The former techniques build a classifier from a labeled training dataset that explicitly distinguishes normal and abnormal behavior (Shon et al., 2005; Lee et al., 1998). On the other hand, unsupervised techniques (semi-)automatically infer baseline behaviors, which is assumed to represent the normal behavior. A variety of techniques have been proposed to address unsupervised anomaly detection, ranging from simple statistical observations (e.g., frequency) on the selected features (Qu et al., 1998; Smaha, 1988) to data mining techniques such as, e.g., clustering (Bolton and Hand, 2001) or Markov chain (Ye et al., 2000).

A well-known drawback of classic anomaly detection systems is that the system behavior is often modeled as a *black box*; namely, the behavior is often encoded into a mathematical model that is either not accessible or, anyway, too abstract and complex to be easily understood by a human analyst (Etalle, 2017). Therefore, analysts are often not aware of what is happening in the system. This poses some challenges in determining how to response upon an alert, thus hampering the *actionability* of anomaly detection systems, which is a crucial property to their application in practice.

In this work, we explore the potentiality of *process analysis* techniques to infer *white box* behavioral model, i.e., models able to represent the end-to-end structure of the system behavior. To this end, we adopt a *process-oriented* perspectives; namely, we assume that activities supported by the system are structured according to an underlying notion of *process*, which poses constraints/guidelines on the order in which activities should be executed, data access and so on. In particular, we adopt a broad notion of process, intended as a set of activities that have to be performed in a given order to achieve a given goal. Adopting a process-oriented perspective allows us to map sets of activities to a specific process execution. This introduces a level of abstraction that provides a comprehensive overview of the system behavior and simplifies the understanding and analysis of behavioral patterns.

The application of process analysis techniques for security analysis is an under-investigated topic. A first step in this direction has been taken in, e.g., (van der Aalst and de Medeiros, 2005; Accorsi et al., 2013). Those works propose to apply a *process discovery* technique on an event log to infer a process model representing the normal system behavior, i.e. the normal ordering relations existing among process activities. However, those works infer a single model of normal behavior from the log. As shown in (Alizadeh et al., 2018b), this can lead to misleading diagnostics. Another stream of research (Adriansyah et al., 2013; Alizadeh et al., 2018a; Genga et al., 2018) propose to detect security threats by identifying deviations of the observed behavior from process specifications. However, these works typically assume that a process model representing the normative behavior is available, which is often not the case in real settings.

Overall, our literature review revealed the lack of systematic approaches able to infer accurate models of normal behavior. In the next section, we present guidelines for a systematic behavioral analysis and discuss the main challenges.

3 METHODOLOGY

In this section, we present a methodology for a systematic process-aware behavior analysis. The main aims consist in *(i)* inferring work practices characterizing the organization processes from the recorded behavior, and *(ii)* analyzing these practices to determine which ones should be deemed representative of the normal system behavior. To this end, first we need to reconstruct behaviors according to the underlying process; then, we have to select features that allows identifying and grouping similar behaviors; finally, we delve into the inferred practices to spot possible security concerns. Next, we present each step in detail.

Behavior identification The first step of the methodology is to represent the logs recording the observed system behavior in a format suitable for the application of process analysis techniques. To this end, the activities recorded in the log should be grouped into process executions. Although several organization IT systems support the generation of this type of logs (e.g., Workflow Management Systems, Enterprise Resource Planning systems), many logging systems are not process-aware; therefore, a preprocessing phase might be needed to identify process executions from the available logs. This preprocessing phase can be further refined in two steps. The first one is aimed to collect and properly integrate process data possibly spread among several and heterogeneous sources (e.g., Excel spreadsheets, or database tables). To this end, *Extract, Transform and Load* (ETL) methodologies and techniques developed within the data warehouse community can be exploited (see, e.g., (Vassiliadis, 2009) for a survey on the most commonly used ETL tools). The second one is devoted to *process identification*. Logging systems that are not driven by a notion of process provide a simple stream of recorded activities, with no indication of which activities belong to the same process execution. Therefore, the event log has to be processed in order to identify and group together activities belonging to a single process execution. It is worth noting that this challenge is far from trivial and several research efforts have been devoted to cope with this issue, e.g. (Ferreira and Gillblad, 2009; Walicki and Ferreira, 2011).

The output of this step is an event log that comprises a collection of *traces*, each corresponding to a specific process execution (also called *case*). Each trace consists of a sequence of *events*, each recording data related to the execution of a process activity.

Behavior profiling The second step of the methodology aims to build *behavioral profiles* from event logs. This involves *(i)* a features selection phase, where relevant features are identified based of the security properties of interest, and *(ii)* a clustering phase, in which behaviors similar with respect to the selected feature set are grouped together.

As regards point *(i)*, one can exploit both properties related to process executions (e.g., their duration) and properties related to single events (e.g., the user who performed the corresponding activities). Although this choice is strongly affected by the data available in the event log, we can list four main general properties that is worth considering for a security analysis: the *control-flow*, which refers to the order in which activities are performed during process executions; the *users*, which refers to the distribution of the workload among involved actors; the *time*, which refers to temporal aspects of process executions (e.g., their durations); and, finally, the *data*, which refers to any other information on the process and/or on the process activities stored in the event log.

These dimensions are often taken into account by process analysis techniques, and we argue that they can provide meaningful insights that can be exploited for the detection of security threats. As an example, analyzing the control-flow one might detect violations of security constraints (e.g., the skipping of some critical checking activities when assessing a loan application); the analysis of users' behaviors might point out undesirable or risky practices (e.g., violations of separation of duty constraints); the analysis of temporal aspects might reveal suspicious differences in the completion time of process executions; and so on.

It is worth noting that feature selection is often an incremental step, especially when little or no knowledge on the underlying processes is available. In these cases, one can proceed by selecting one dimension per time and checking the obtained behaviors, then using the obtained findings to further refine the elicited features.

Once the relevant features have been identified, behaviors clustering is performed to build *behavioral profiles*, i.e. clusters of behaviors that are similar according to the selected feature set. Behaviors clustering requires determining suitable ranges for the set of intervals of the selected features. Intervals can be determined in several ways. A simple solution consists in exploiting simple statistics, e.g. mean, median, quartiles, to manually determine the intervals; while a more sophisticated ones consists in adopting automatic *trace clustering* techniques (Hompes et al., 2015; Song et al., 2008). The choice of the technique depends on the set of features and on the process at hand. For instance, if a single feature is used for clustering, the use of simple statistics might be sufficient to detect meaningful intervals; on the other hand, if the combination of several properties is used, trace clustering techniques offer a more effective alternative.

The output of this step consists in a set of behavioral profiles, each representing a cluster of behaviors similar with respect to a given feature set.

Behavior Analysis The final step of the methodology focuses on exploring the obtained behavioral profiles to determine which ones should be considered representative of *normal* behavior and which ones should not.

When no a-priori knowledge is available, a common practice to determine what behavior should be considered as normal is to use statistical measures (Alizadeh et al., 2018b). In fact, it is reasonable to assume normal behavior to be the one occurring more frequently in the event log. Therefore, at first one might label as normal behavior the one corresponding to the interval covering the standard deviations, when the profiling is performed by means of statistics on single features, or the one corresponding to the largest cluster, when trace clustering techniques are applied.

However, we argue that this simple labeling is not enough to obtain a proper comprehension of the process and further investigation should be performed. For example, it might be the case that a behavior, considered normal with respect to a given feature, can actually raise security concerns from another perspective; or, on the opposite, behaviors that are less frequent or, anyway, diverge from the normal distribution, do not reveal particular security concerns, so that they could be labeled as normal as well.

To address this issue, as a general guideline, we propose to analyze behavioral profiles in order to determine which ones correspond to desirable working practices and which ones do not. To this end, besides analyzing behavioral models individually, we also propose to compare the obtained behavioral profiles with each other. As shown in the next section, this analysis indeed provides more in-depth understanding and insights of working practices.

4 CASE STUDY: BPI2012 EVENT LOG

In this section, we present an application of the proposed methodology to a real-life event log. We first introduce the dataset; then, we present the findings obtained by inferring and exploring behaviors from the log.

4.1 Dataset

We used the event log recording the loan management process of a Dutch Financial Institute, which was made available for the 2012 BPI challenge (BPI Challenge 2012, 2012). The event log contains the events recorded for three intertwined subprocesses: subprocess A specifies how loan applications should be handled, subprocess O describes how loan offers should be handled, and subprocess W specifies how work items are processed. For each activity, the log stores its *name*, *timestamp* and the involved *user*. Moreover, for each application the log provides the *requested amount*. The first row of Table 1 shows some statistics of the event log, namely the number of traces, activities and events, together with the minimum, maximum and mean number of events in each trace.

Table 1: Statistics related to the entire set of traces and to the traces corresponding to approved applications.

Behavior	**Cases**	**Activities**	**Events**	**Min Events**	**Max Events**	**Mean Events**
all	13087	36	262200	3	175	20
approved	2246	21	99925	22	163	44

4.2 Behavior Identification

The BPI2012 log consists of traces grouping activities belonging to the same process execution; thus, a preprocessing step is not needed for this log. Nevertheless, analyzing this log, we observed a strong variability among process traces. In particular, the log records process executions related to approved applications, declined applications, canceled applications, and running applications. It is reasonable to expect that traces concerning the approval of applications differ significantly from traces concerning, e.g., the denial of applications. Such heterogeneity usually has a negative effect on the results of process analysis techniques.

Among them, we decided to focus on a subset of the traces, i.e. those related to *approved* applications. This is because, from a security viewpoint, approved applications are more critical than others, since security incidents in those process executions can result in a financial loss for the financial institute. Process executions concerning the approval of applications are typically characterized by the occurrence of activity *A_APPROVED* and the absence of activities *A_DECLINED*, *A_CANCELED*. We filtered the log accordingly and obtained a new event log. The second row of Table 1 shows statistics of the resulting event log. There are 2246 approved applications, corresponding to approximately 17% of the total number of applications.

4.3 Behavior Profiling

Behavioral profiles can be built with respect to the dimensions discussed in Section 3. For the sake of space, in this work we present the results for only one dimension, namely the *temporal duration* of process executions. We selected this feature since large differences in the duration of process executions might indicate differences in the conscientiousness applied in the approval of application, which is in general undesirable.

From the event log we observed that the mean time of approval is around 17 days; to delve more into the temporal dimension, we plot histogram related to applications completion times in days, reported in Figure 1. The distribution shows that applications are mainly closed before 40 days; analyzing the event log,

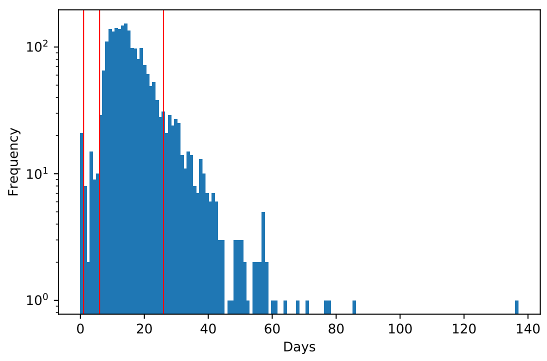

Figure 1: Histograms of temporal duration of approved applications in logarithmic scale.

we observed that most of the applications are closed within 10-30 days. However, there are some notable outliers, some of which required much more time. It is also worth noting that we even have a set of applications closed in less than one day.

These differences in the duration of process executions confirm that the temporal dimension is indeed relevant for this dataset. If on the one hand these differences might simply be due, for instance, to different customers (e.g., the procedure is likely to be much smoother and shorter for well-known customers than for new ones); on the other hand, they might also be due to shallow checking or even abuses. Since we are considering a single feature, we exploited some basic statistics for behavior clustering. The mean duration time for approved applications is 16.8 days, with a standard deviation of 9.75. Therefore, we marked as *normal* behavior all applications approved in more than six days and in less than 27 days. Applications longer than 27 days are marked as *long* applications. We partitioned shorter applications in two classes: *short* applications, i.e. applications approved in more than 1 day and in less than 6, and *instant* applications, i.e. applications that were approved in few hours and hence result of particular interest for the scope of our analysis. Red lines in Figure 1 delimit the portion of applications belonging to each class (e.g., the first line delimits the portion related to instant applications, the second line short applications and so on). Table 2 shows some statistics for the logs corresponding to the classes of behaviors considered in our analysis. Note that other statistics, as well as trace clustering techniques, could been used as well to discriminate behaviors.

4.4 Behavior Analysis

The last step of the methodology in Section 3 involves a comparative analysis of the obtained behavioral profiles to support analysts in characterizing *normal* behaviors from abnormal ones. Next, we show the results of the comparison considering control-flow, time, users and loan amount as features.

Table 2: Statistics of the event logs related to approved applications.

Behavior	Cases	Activities	Events	Min Events	Max Events	Mean Events
long	292	21	20059	26	163	69
normal	1877	21	77284	22	109	41
short	50	20	1855	22	70	37
instant	27	19	727	22	49	27

Control flow The first feature we consider for the comparison is the control-flow of the behavior clusters. By comparing the control-flow we expect to reveal differences in whether and how activities are performed within each cluster. In particular, we can determine, for example, whether some checking activities tend to be delayed or completely skipped in some cluster.

To determine the control-flow, we need to uncover the process *structure* from the event log. To this end, we employ *process discovery* techniques, which analyze the ordering relations among events stored in the event log to infer a process *model*. Since our process is rather complex and involves many different variants, we apply a heuristic technique that only highlights the most relevant process behaviors. More precisely, we use Disco[1], a well-known tool for process discovery.

Given an event log, Disco returns a process model in which each node corresponds to a process activity and each edge represents an ordering relation between a pair of activities. Only activities and relations whose frequency is above a user-defined threshold are reported. For our analysis we used the default settings. Figure 2 shows the process model returned by Disco for long applications. The numbers inside the boxes and above the edges show the maximum number of repetitions within a trace of the corresponding activity/relation. On the edges it is also displayed, in lighter black, the average time between the pair of activities.

The discovered model highlights three main phases of the process. The top one involves some initial activities concerning the submission of the application and the starting of the process; the central part involves activities related to the check of the application and to the creation and the negotiation of an offer with the applicant; finally, the bottom part comprises additional controls on the application,

[1]https://fluxicon.com/disco/

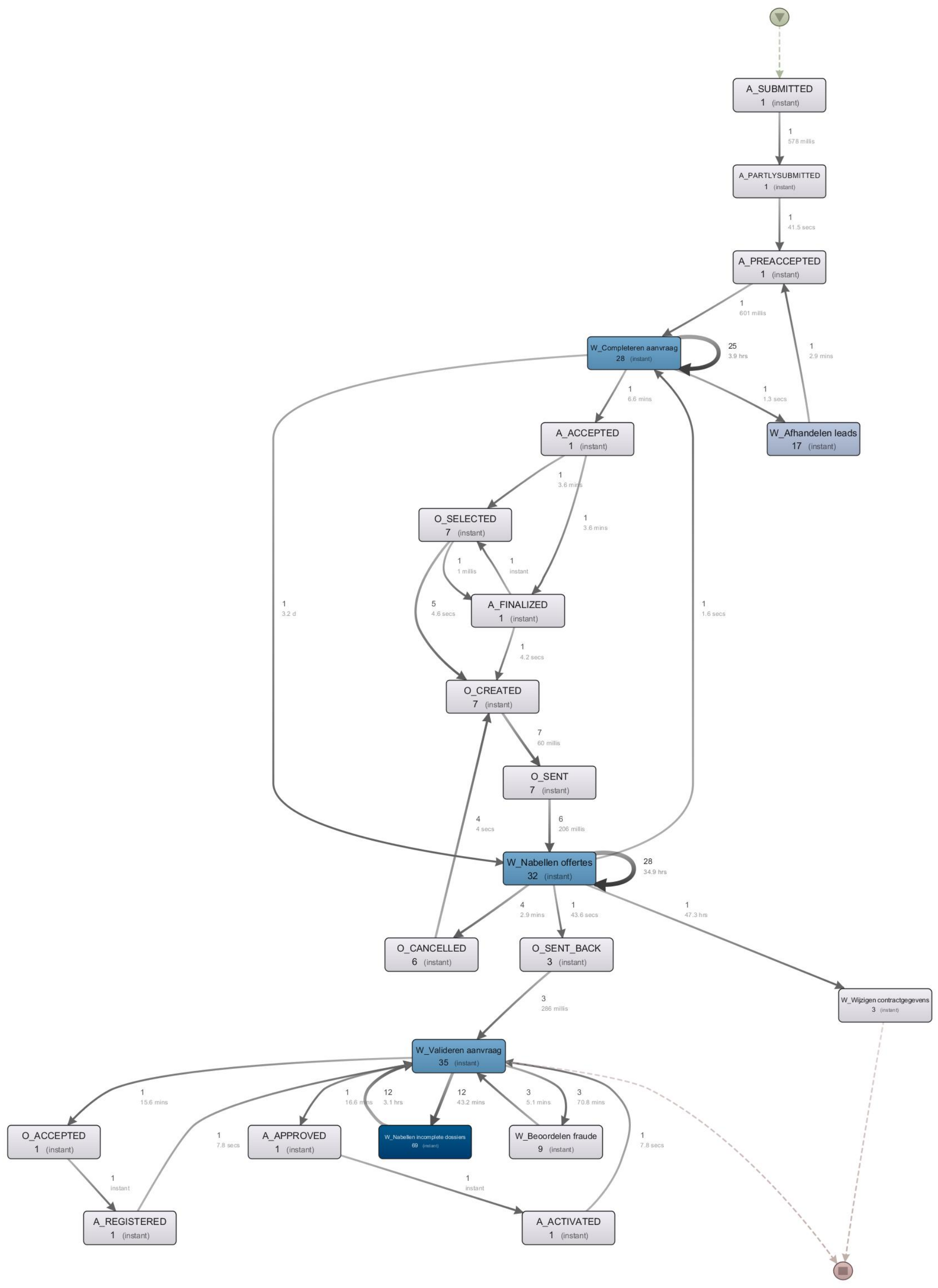

Figure 2: Activities flow of long applications.

e.g. fraud checking, together with the finalization of the application. For the other types of applications (Table 2), we found almost the same structure, with some minor variations (e.g., some missing activities in shorter applications). The main differences lie in the number of times activities are repeated, especially concerning activities related to application check, and in the average times between pairs of activities. For example, activity *W_Completerenaanvrag* was repeated a maximum of 7 times within *instant* applications, against a maximum of 28 for *normal* applications. Moreover, the average time between activities *W_Completerenaanvrag* and *W_Nabellenoffertes* is less than two minutes, against

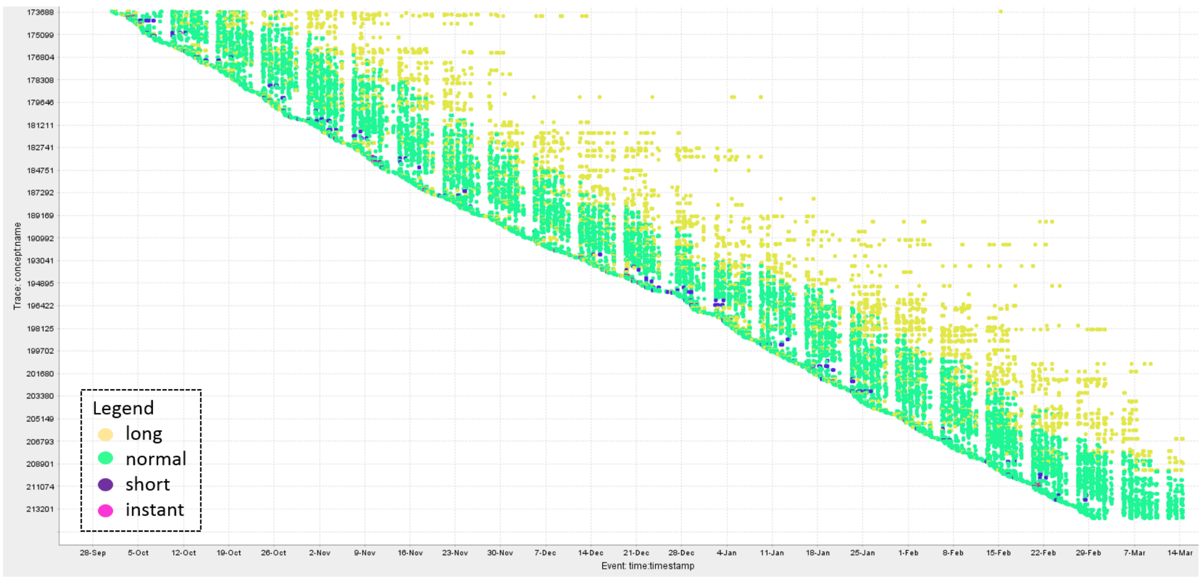

Figure 3: Workload distribution over time for approved applications.

an average time of 3 days for the normal cases. This implies that *instant* applications were characterized by few rounds of checking and a very quick interaction with the applicant.

Another interesting difference is that some process activities have not been performed in shorter applications (i.e., *short* and *instant*). In particular, in those applications activity *W_Wijzigencontractgegevens* was not executed, indicating that there were no changes made to the contract after the approval. In addition, activity *W_Beoordelenfraude* was not performed in *instant* applications, indicating that a check for possible frauds was never performed for these applications.

Time The second features we take into account for our comparison is the temporal distribution of activities. In particular, we intend to analyze how activities in different behavioral profiles are distributed among working days and week-ends. Since the BPI2012 log concerns a financial institute, we expect activities to be executed only from Monday to Saturday; activities performed on Sunday would represent an unusual, possibly suspicious, behavior that needs to be checked.

Figure 3 displays a dotted chart showing the activities performed within each process execution against the corresponding days, thus allowing us to explore process workload over time. The x-axis represents the events' timestamps, whereas the-y axis the trace identifiers. Colors represent classes of behaviors.

We can observe that process activities of long and normal applications are mainly grouped in bunches, separated by inactivity periods. Checking the corresponding timestamps, we observed that these interruptions correspond to week-ends. In this respect, it is worth noting that, although the workload is much less intense, many activities were also performed during the week-end. We could not find a similar case for the short and instant applications. However, we observed some regularities in instant applications, i.e. they tended to occur in groups of three or four.

Amount We also analyzed the distribution of the requested amounts for each behavioral profile to determine whether there exist a correlation between them. It is reasonable to expect to find some correlation between the requested amounts and the duration of the process since higher amount should require more careful controls and negotiations with the applicant. In general, the approval of applications requesting high amounts in short period of time could be undesirable and might deserve further investigations to understand the reasons.

The results of the analysis are reported in Figure 4. It is worth noting that the distribution of requested amounts look roughly the same for the different behavioral profiles, with the exception of some high values that occur mostly in the normal and long applications (marked as outliers in the figure). Even more surprisingly, it looks like *instant* applications involved amounts overall higher than amounts requested in other applications. Nonetheless, it is worth noting that we have less cases for this type of applications, which means that variations of even few cases have a larger impact on the overall distribution.

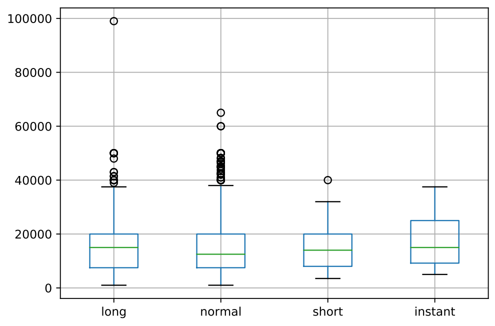

Figure 4: Distribution of the amount requested for each process behavior.

To confirm that this difference is statistically not significant, we also performed a hypothesis testing, comparing the distribution of the amounts requested in normal applications with the distribution of the amount requested in the other classes of applications. Since the distributions of amounts do not fit the normality assumption, we adopted a non-parametric test. In particular, we used the WilcoxonMannWhitney two-sample rank-sum test (Mann and Whitney, 1947), which is a test commonly used when the assumptions of t-test are not met.

Note that we have a sufficient number of samples to perform this test for almost all pairs.[2] The only exception is the comparison between the normal and instant applications; nevertheless, we have 27 applications for the latter, so the results we obtain can be still considered a good approximation.

For the testing between normal and long applications, we obtained a p-value of 0.28; between normal and short applications we obtained a p-value of 0.69; finally, between normal and instant applications we obtained a p-value of 0.40. Since differences between distributions are typically considered of statistically significance when the p-value is smaller than 0.1, the obtained values confirm that the differences we observed for the amounts requested in the different classes of applications are not statistically relevant.

Users The process at hand involves several, and often iterated, activities related to the management and checking of applications. Therefore, by analyzing the workload distribution we can grasp some insights on whether duties tend to be distributed among different actors. Such information can help analysts in understanding whether the responsibilities are properly distributed among users. As an example, one can check whether critical activities (e.g., the approval of an application) are performed by different users, or, instead, few users performed most of the critical activities. The latter is an undesirable situation since it might indicate violations of separation of duty constraints.

[2]Common rule-of-thumb for the applicability of Wilcoxon-Mann-Whitney test is to have at least 30 samples per group.

Table 3: Workload distribution over users for approved applications.

Behavior	Min Actors	Max Actors	Mean Actors
long	5	28	12.5
normal	3	20	7.3
short	3	10	5.3
instant	2	5	2.6

We analyzed the number of different users involved in each process execution, computing the minimum, maximum and average number for each behavioral profile. Results are reported in Table 3. It is worth noting that different behavioral profiles are characterized by a different workload distribution. In particular, while longer applications seem to normally involve many different actors in their executions, much less actors are normally involved in the approval of shorter applications. This is especially evident for *instant* applications, in which two to five employees always performed of all process activities.

5 DISCUSSION AND CONCLUSION

In this work, we performed a preliminary exploration of the potentialities of process-aware behaviors analysis for security. We adopted a process-oriented perspective to structure raw data recorded by logging systems. By exploiting this representation, we delineated a methodology to build accurate and understandable behavioral profiles that can assist analysts in the understanding of the current work practices. We applied the proposed methodology to a real-world event log, providing some concrete examples of how off-the-shelf process analysis techniques and tools can be used to explore several security-related aspects of process executions along different dimensions.

Despite our analysis showed that process-aware behavioral analysis has great potentialities when applied to security, it presents a number of challenges. Next, we list some of the most relevant ones:

- **Data Collection & Preprocessing:** Our analysis assumes that it is possible to obtain an event log grouping all activities related to the same process execution. Although the log used in our case study meets this demand, this is not always the case. In many IT systems, log data might be spread along several and heterogeneous data sources (e.g., Excel spreadsheets, databases). This requires retrieving all scattered data and merging them in a proper format for the analysis. Even when events have been reconstructed along with the relevant information, additional challenges have to be face to identify which events belong to the same process execution (Alizadeh et al., 2018b). Solving such challenges is far from being trivial.

- **Features selection:** The choice of the features to be considered depends on the application domain and scope of the analysis. In particular, it requires background knowledge of the underlying process and prior knowledge of what to look for, which is not always the case, especially in the security context. On top of this, the analysis is constrained by the information available in the log.
- **Technique choice:** There is not a one-fit-all technique for all cases. The choice of the techniques to be used for the analysis depends on the scope of analysis and type of data. This requires experienced and highly skilled analysts, with a strong background both in security and in data analysis techniques.

More research efforts are necessary to explore and systematize findings and results obtained so far and to develop a more general framework. In future work, we plan to investigate these issues. In particular, we intend to further elaborate on the observations made in this work to devise general guidelines to apply data science to behavior analysis for security, taking into account a larger range of techniques. At the same time, we plan to perform an extensive experimental evaluation on both synthetic and real-world logs.

ACKNOWLEDGMENTS

This work is partially supported by ITEA3 through the APPSTACLE project (15017) and by ECSEL through the SECREDAS project.

REFERENCES

Accorsi, R., Stocker, T., and Müller, G. (2013). On the exploitation of process mining for security audits: the process discovery case. In *Proceedings of Annual ACM Symposium on Applied Computing*, pages 1462–1468. ACM.

Adriansyah, A., van Dongen, B. F., and Zannone, N. (2013). Controlling break-the-glass through alignment. In *Proceedings of International Conference on Social Computing*, pages 606–611. IEEE.

Alizadeh, M., Lu, X., Fahland, D., Zannone, N., and van der Aalst, W. M. P. (2018a). Linking data and process perspectives for conformance analysis. *Computers & Security*, 73:172–193.

Alizadeh, M., Peters, S., Etalle, S., and Zannone, N. (2018b). Behavior Analysis in the Medical Sector: Theory and Practice. In *Proceedings of ACM/SIGAPP Symposium On Applied Computing*. ACM.

Association of Certified Fraud Examiners (2018). Report to the Nations: 2018 Global study on occupational fraud and abuse. `http://www.acfe.com/report-to-the-nations/2018/`. Accessed: 2018-06-01.

Bolton, R. J. and Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. In *Proceedings of Credit Scoring and Credit Control*, pages 235–255.

BPI Challenge 2012 (2012). Event log of a loan application process. `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`.

Cao, L. (2010). In-depth behavior understanding and use: The behavior informatics approach. *Information Sciences*, 180(17):3067 – 3085. Including Special Section on Virtual Agent and Organization Modeling: Theory and Applications.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15.

Costante, E., Fauri, D., Etalle, S., den Hartog, J., and Zannone, N. (2016). A hybrid framework for data loss prevention and detection. In *Proceedings of IEEE Security and Privacy Workshops*, pages 324–333. IEEE.

Etalle, S. (2017). From intrusion detection to software design. In *Computer Security*, pages 1–10. Springer.

Ferreira, D. R. and Gillblad, D. (2009). Discovering process models from unlabelled event logs. In *Business Process Management*, pages 143–158. Springer.

Genga, L., Alizadeh, M., Potena, D., Diamantini, C., and Zannone, N. (2018). Discovering anomalous frequent patterns from partially ordered event logs. *Journal of Intelligent Information Systems*.

Hompes, B., Buijs, J., van der Aalst, W., Dixit, P., and Buurman, J. (2015). Discovering deviating cases and process variants using trace clustering. In *Proceedings of Benelux Conference on Artificial Intelligence*, pages 5–6.

Lee, W., Stolfo, S. J., et al. (1998). Data mining approaches for intrusion detection. In *Proceedings of USENIX Security Symposium*, pages 79–93.

Mann, H. B. and Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Ann. Math. Statist.*, 18(1):50–60.

Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470.

Qu, D., Vetter, B. M., Wang, F., Narayan, R., Wu, S. F., Hou, Y., Gong, F., and Sargor, C. (1998). Statistical anomaly detection for link-state routing protocols. In *Proceedings of International Conference on Network Protocols*, pages 62–70. IEEE.

Richardson, R. (2008). CSI Computer Crime and Security Survey.

Shon, T., Kim, Y., Lee, C., and Moon, J. (2005). A machine learning framework for network anomaly detection using SVM and GA. In *Proceedings from Annual IEEE SMC Information Assurance Workshop*, pages 176–183.

Smaha, S. E. (1988). Haystack: An intrusion detection system. In *Proceedings of Aerospace Computer Security Applications Conference*, pages 37–44. IEEE.

Song, M., Günther, C. W., and van der Aalst, W. M. (2008). Trace clustering in process mining. In *Business Process Management*, pages 109–120. Springer.

van der Aalst, W. M. and de Medeiros, A. K. A. (2005). Process mining and security: Detecting anomalous process executions and checking process conformance. *Electronic Notes in Theoretical Computer Science*, 121:3–21.

Vassiliadis, P. (2009). A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining*, 5(3):1–27.

Walicki, M. and Ferreira, D. R. (2011). Sequence partitioning for process mining with unlabeled event logs. *Data & Knowledge Engineering*, 70(10):821 – 841.

Ye, N. et al. (2000). A markov chain model of temporal behavior for anomaly detection. In *Proceedings of IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169.