Using Process Mining and Model-driven Engineering to Enhance Security of Web Information Systems

Simona Bernardi
Centro Universitario de la Defensa
Zaragoza, Spain
Email: simonab@unizar.es

Raúl Piracés Alastuey, Raquel Trillo-Lado
Universidad de Zaragoza
Zaragoza, Spain
Emails: raul.piraces@gmail.com, raqueltl@unizar.es

***Abstract*—Due to the development of Smart Cities and Internet of Things, there has been an increasing interest in the use of Web information systems in different areas and domains. Besides, the number of attacks received by this kind of systems is increasing continuously. Therefore, there is a need to strengthen their protection and security. In this paper, we propose a method based on Process Mining and Model-Driven Engineering to improve the security of Web information systems. Besides, this method has been applied to the SID Digital Library case study and some preliminary results to improve the security of this system are described.**

***Keywords*-process mining; model-driven engineering; security; web information systems**

I. Introduction

Since the last decades, most institutions and companies have opted for the use of Web applications due to their advantages over traditional desktop applications (cross-platform compatibility, interoperability, accessible anywhere from a wide range of devices, etc.). The bigger their adoption of Web applications is, the more attractive they become to people related to cybercrime. Thus, according to the security reports by Symantec [1], the number of attacks to Web information systems in 2015 is the double as in 2014.

Web information systems are usually composed of two main components: the *front-end*, or presentation layer, and the *back-end*. The *front-end* is composed of multiple resources (html templates, images, CSS and JavaScript files, etc.) which are accessed by means of the users' browsers and provide the main interface of the system; while the *back-end* provides different functionalities or services and, generally, it is not directly accessed by the final users. Currently, there exists a wide set of technologies and languages to develop Web information systems such as PHP, .Net, Java Enterprise Editions (JEE), Ruby on Rails, etc. Nevertheless, all these types of Web applications are based on similar principles and use the stateless Hyper Text Transfer Protocol (HTTP).

In this paper, we propose a method based on Process Mining [2] and Model-Driven Engineering [3] to improve the security of Web information systems that does not depend on a specific technology. In particular, the method supports the detection of attack patterns by identifying the deviations from the known system behavior, which is represented by a *normative* model. This model is automatically generated from the UML specification of the system, by applying a model transformation technique, and it is the input of a trace-driven simulator, together with the logs produced by the system in operation, to identify deviations. Such deviations are further analyzed by using the fuzzy mining discovery technique to detect attacks to the system. Besides, in this paper, we also describe how this method has been used to analyze the logs of the last ten years of the Web information system SID Digital Library[1] and provide some preliminary results to improve the security of that system.

The paper is organized as follows. Firstly, related work is analyzed in Section II. Secondly, an overview of the proposed method, describing its main steps, is presented in Section III. Details of the main steps and how they have been performed to analyze the logs of the SID BiD are presented in Sections IV, V and VI. Finally, conclusions and future work are given in Section VII.

II. Related work

Traditionally, information systems have been protected by means of preventive and reactive approaches to intrusion detection [4], which mainly rely on data mining and machine learning techniques [5]. Preventive approaches establish rules in Security Information and Event Management (SIEM) systems to reduce or remove the success conditions of cyberattacks. Rule based detection via known signature is the most common approach, where the analysis techniques compare data against a set of signatures of attack features. This approach is simple to implement, however it can be applied when the features of an attack pattern are known. On the other hand, reactive approaches are based on anomaly detection techniques that analyze the behavior of the system in operation and look for deviations from what it is considered to be *normal*. Reactive approaches are therefore more suitable than the former to detect attacks which exploit unknown vulnerabilities, such as the *zero-day attacks*. The

[1]SID BiD is a Web information system developed with Java technologies available at http://sid.cps.unizar.es/BiD/.

main disadvantage is the generation of false positives and false negatives.

The method presented in this paper is a mixed approach (preventive-reactive) since it supports the detection of undocumented threats in Web information systems and the enhancement of their security by adding new detection rules for such threats. The proposed method is based on the process mining discipline [2] which provides techniques for *process discovery*, i.e., deriving process models from logs, *conformance checking*, i.e., checking the alignment of an existing/derived process model and logs, and *process enhancement*, i.e., enriching the process model through mining additional perspectives such as timing. The data input for process mining may come from different sources and needs to be pre-processed to get *event logs*, which are structured data related to ordered events occurring within a process. Process mining mainly addresses business processes and, to the best of our knowledge, few proposals [6], [7] apply it to solve security issues in the information system domain. In particular, the work [6] explores the potential capabilities of discovering techniques for anomaly detection in security audit of business processes. Concrete discovery algorithms are instead proposed in [7] to detect anomalies in information systems: such algorithms mainly focus on the detection of infrequent execution traces.

III. Method overview

The proposed method consists of five main phases or steps (see Figure 1):

- *Step 1*. Specification of the system behavior by means of the Unified Modeling Language (UML) [8].
- *Step 2*. Automatic generation of a formal model from the UML-based specification that we call *normative* model.
- *Step 3*. Control and monitoring of the Web information system to get data logs that represent its operative (or real) behavior.
- *Step 4*. Pre-processing of the logs of the system to get *event logs*, which can be analyzed using process mining techniques.
- *Step 5*. Identification of deviations between the normative model and the operative behavior by means of different process mining techniques.

Notice that Steps 1 and 2 (i.e., obtaining the normative model) can be carried out in parallel with Steps 3 and 4 (i.e., obtaining event logs). Nevertheless, Step 2 and Step 4 must have finished in order to execute Step 5 (the analysis and comparison between the expected and real behavior of the system).

The detected deviations can represent anomalies (attacks) or, simply, new use case scenarios that were not considered in the initial specification. When a new use case scenario is discovered, the initial UML diagrams are enriched to include it and avoid detecting deviations in those cases. On

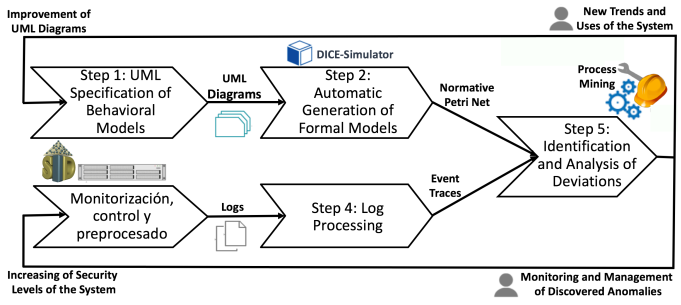

Improvement of UML Diagrams
New Trends and Uses of the System
Step 1: UML Specification of Behavioral Models
UML Diagrams
DICE-Simulator
Step 2: Automatic Generation of Formal Models
Normative Petri Net
Process Mining
Step 5: Identification and Analysis of Deviations
Monitorización, control y preprocesado
Logs
Step 4: Log Processing
Event Traces
Increasing of Security Levels of the System
Monitoring and Management of Discovered Anomalies

Figure 1. Overview of the proposed method.

the other hand, if the deviation represents a potential attack, then, system administrators must adapt the configuration of the Web server or other Security Information and Event Management (SIEM) systems in order to mitigate or remove the conditions on which that potential attack can happen.

These steps have been applied to the SID Digital Library, a Web information system developed by the SID research group of the University of Zaragoza to manage the publications of its members and other researchers from research groups that usually collaborate with SID members, such as the BDI research group[2] from the University of Basque Country and the OEG research group[3] from the Technical University of Madrid. The SID Digital Library was developed in 1997. Nevertheless, it was completely updated and re-implemented in 2010. Since then, it has had little updates. Currently, this system offers 14 uses cases (insert publication, search for publications with different criteria, edit references, etc.) and it is used by around 35 different researchers to manage their publications. Besides, anonymous users use this system to obtain information about papers published by researchers from SID, BID and OEG.

IV. Getting a normative model

The first two steps of the method concern the definition of the system expected (or known) behavior and the derivation of a formal model from the system specifications. This formal model is named *normative* model, and can be analyzed by using process mining techniques. To this aim we apply model transformation approaches in the literature, that propose the generation of Petri Net models from UML activity diagrams [9] or sequence diagrams [10]. Although the correctness of the transformations have not been proved formally, they rely on mature approaches and have been empirically validated with an extensive set of case studies.

Since the system is already in operation, we could assume that UML-based behavioral models are already available. However, when the maintenance of the systems and the management of the life cycle of software is not appropriate, this kind of models can be missed. In these cases,

[2]BDI research group website: http://bdi.si.ehu.es/bdi/.
[3]OEG research group website: http://mayor2.dia.fi.upm.es/oeg-upm/.

these models can be obtained from the implementation of the system by using reverse engineering techniques and/or refining the design specifications. For example, the initial UML behavioral models used to develop SID BiD were not available when we started this study.

The UML specification can represent a partial behavioral view of the system: in the SID BiD system, we consider the UML activity diagram shown in Figure 3, that models the scenarios related to a subset of the use cases of Figure 2 (in particular, the grey filled ones), where each action represents an HTTP request from the user.

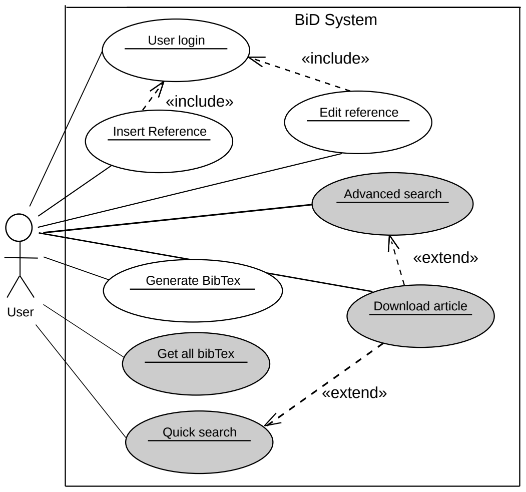


Figure 2. BiD system main functionalities.

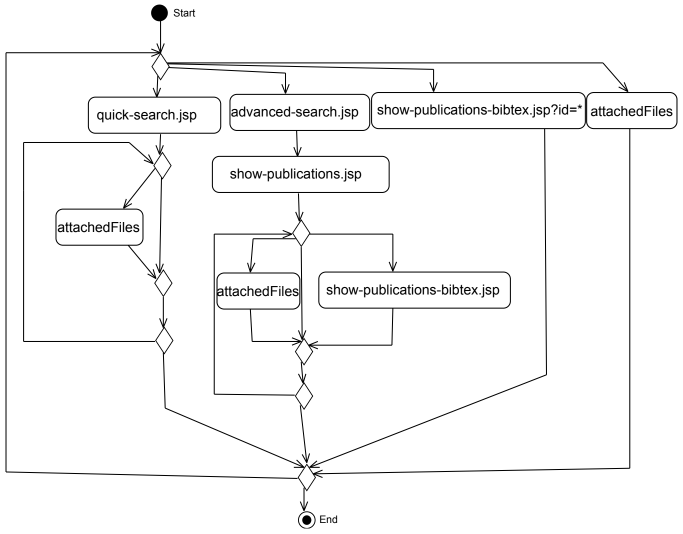


Figure 3. BiD system behavioral view (partial).

The Petri Net model of Figure 4 is the normative model of the SID BiD system and it has been automatically generated from the Activity diagram of Figure 3 by using the DICE Simulation tool [11]. The latter implements the model transformations in [9], [10], and generates a Petri Net model in the PNML standard format [12]. In the figure, the labeled transitions correspond to the homonym actions in the Activity diagram, while the rest of transitions model the control flow. The initial marking of the Petri Net model represent the initial state (*Start*) of the activity diagram.

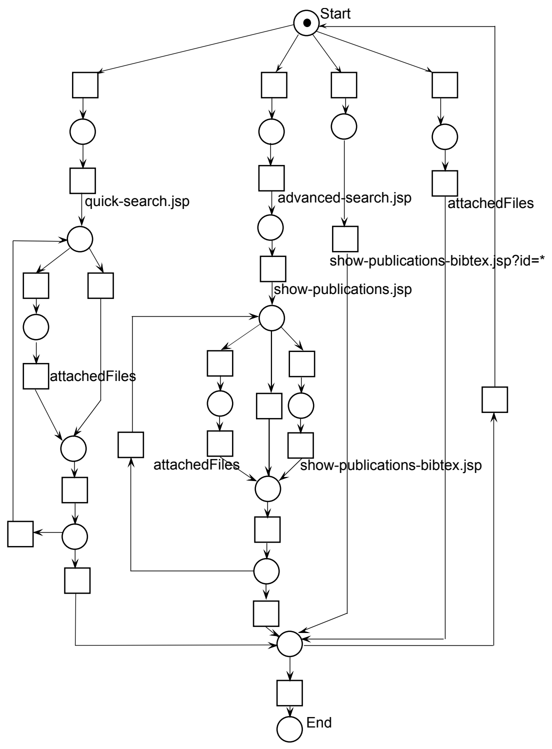


Figure 4. Petri Net normative model.

V. Log processing

In order to apply the different process mining techniques enumerated in Section II (*process discovery*, *conformance checking* and *process enhancement*) and study the real behavior of the system, extraction and analysis of the *event logs*[4] from the different components of the system are key processes (as event logs are considered evidences of events happening in the system while it is running). Due to the fact that Web information systems involve a great number of components and technologies (e.g., Web servers, Data Bases Management Systems, File Systems, etc), the integration of the information coming from the different logs is needed before applying process mining techniques. Nevertheless, in our case of study (the SID BiD Web information system) only logs about the Web server (in particular, from the Web server Apache Tomcat) were available. So, no integration

[4]An event log of a component of an information system is a basic resource (e.g., a file or a database) that registers and stores information about the different events happening in the component.

of information coming from different components (data sources) was needed.

Moreover, to apply process mining techniques an event in an event log must refer to:

- *A case*. A case is a particular execution of a process, thus each event refers to a trace (or sequence) of related events associated to the execution of a process.
- *An activity*. An activity is an action or task to be performed in a process (as a process is a set of coordinated activities or actions to achieve a particular goal).
- *A timestamp*. A timestamp is a mark indicating when the event happened.
- *Other information*. Other specific data or information related to the execution of the activity.

In our case of study, each entry of the Apache Tomcat logs contains a timestamp and refers to an HTTP request of a particular resource or URL (which we consider as the activity identifier). Besides, in order to assign a case identifier (case-id) to the entries available in the logs used in the experiments, we considered the following options to study the real behavior of system:

1) The work sessions of the users. This work sessions are studied to analyze how a particular user interacts with the system in a work session.
2) The use cases executed by a user. This parameter is used to analyze how a particular user interacts with the system in a work session when she wants to achieve the goals associated to a particular use case.

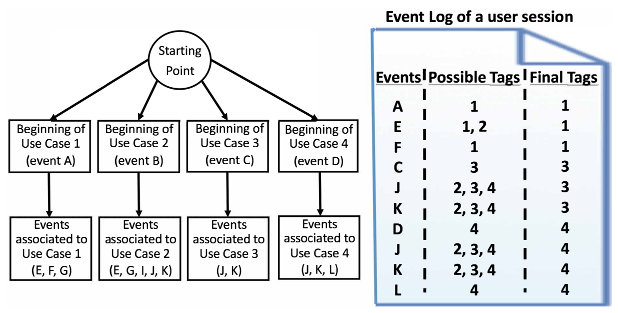
However, the available logs for the study did not contain fields referring to either user sessions or use cases of the SiD BiD Web information System. So, different heuristics were considered to obtain that information from the logs:

- *Heuristic to identify the user session (HSession).* Most web servers have mechanisms to manage user sessions[5] in Web applications, as HTTP is a stateless protocol. Thus, this is a parameter that can be available on the log entries of most Web servers. However, the user session identifier was not available in the logs that were considered for the experiments with SID BiD Web information System. So, a strategy to tag the entries with a user session identifier was developed. In particular, we consider that two entries belong to the same user session if and only if:
  1) they come from the same IP address,
  2) the time elapsed between the two entries is less than a certain threshold (in our case we considered 30 minutes),
  3) they have been performed by the same user if the studied web application allows different users, and
  4) they come from the same browser.

[5]A user session (generally identified by a session id) groups the activities that a user spends on a Web site during a specified period of time.

- *Heuristic to identify the use case performed by a user (HUseCase).* In the context of a software application, a use case is a set of possible actions or events defining how a user interacts with the system to achieve a certain goal; for instance, obtaining a list of papers that satisfy a set of search criteria (published by a particular author, in a specific year, etc.) is a use case of the SID BiD Web information system (cf. the "Advanced Search" use case in Figure 2). In general, several users may interact concurrently with the system to achieve the goals defined by the use cases. However, the entries of the logs of Web servers do not include a specific attribute that relates log entries to the use cases being executed by the users. Therefore, we consider two possible strategies:
  1) updating the code (the implementation) of the Web information systems in order to include the use case identifier being executed in the logs of the Web server, and
  2) tagging the logs entries with use case identifiers.

  The former provides precise results but may be not feasible (for example, if the source code of the Web information system is not available); while the latter may introduce some errors in the tagging process, i.e., some log entries could be associated to an incorrect case use identifier. In the study performed with SID BiD, the tagging process approach was performed, due to the fact that obtaining the source code of the Web information system SID BiD was not possible. In particular, two options were evaluated to tag the log entries with an appropriate case use identifier:
  1) Tagging based on the entry Web pages (or URL) of the use cases. In this case, we consider that an instance of a specific use case starts when an HTTP request for its entry Web page appears in the log and finishes when other HTTP request for other entry Web page (corresponding to another use case) appears (see Figure 5 for an example). This option was discarded due to the fact that several use cases in SID BiD have the same entry Web page; for instance, "Insert Reference" and "Insert Reference as BibTEX register" have the same entry Web page.
  2) Tagging based on the longest path of consecutive activity events of a user. Firstly, in the setting configuration phase of the proposed method, every use case is associated to a list containing all possible actions or events in its execution, i.e., a use case is associated to the list of possible HTTP requests performed when a user is interacting with the system to achieve the goal associated to that use case. After that, each log entry is tagged with a use case identifier by considering in which

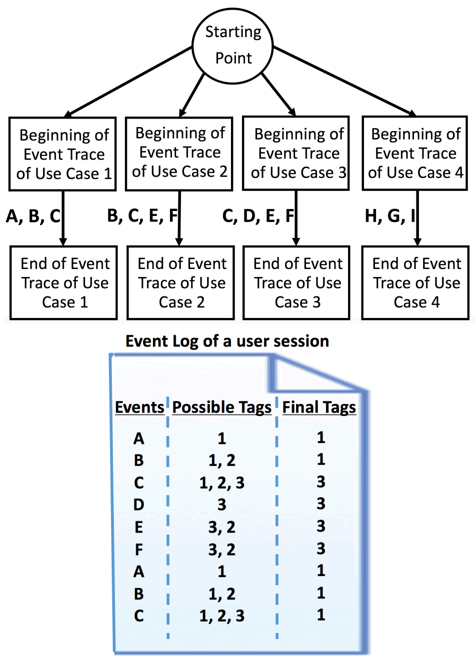

Starting Point
Beginning of Use Case 1 (event A)
Beginning of Use Case 2 (event B)
Beginning of Use Case 3 (event C)
Beginning of Use Case 4 (event D)
Events associated to Use Case 1 (E, F, G)
Events associated to Use Case 2 (E, G, I, J, K)
Events associated to Use Case 3 (J, K)
Events associated to Use Case 4 (J, K, L)

Event Log of a user session

Events	Possible Tags	Final Tags
A	1	1
E	1, 2	1
F	1	1
C	3	3
J	2, 3, 4	3
K	2, 3, 4	3
D	4	4
J	2, 3, 4	4
K	2, 3, 4	4
L	4	4

Figure 5. Tagging based on the entry Web pages of the use cases.

use cases is possible the request corresponding to the log entry. If several options are possible, the option that provides the "longest path of consecutive events" is chosen, i.e., the use case associated to more consecutive actions or events is selected (see Figure 6 for an example).


Starting Point
Beginning of Event Trace of Use Case 1
Beginning of Event Trace of Use Case 2
Beginning of Event Trace of Use Case 3
Beginning of Event Trace of Use Case 4
A, B, C
B, C, E, F
C, D, E, F
H, G, I
End of Event Trace of Use Case 1
End of Event Trace of Use Case 2
End of Event Trace of Use Case 3
End of Event Trace of Use Case 4

Event Log of a user session

Events	Possible Tags	Final Tags
A	1	1
B	1, 2	1
C	1, 2, 3	3
D	3	3
E	3, 2	3
F	3, 2	3
A	1	1
B	1, 2	1
C	1, 2, 3	1

Figure 6. Tagging based on the longest path of consecutive activity events.

Finally, we considered the following case-ids to study the real behavior of system: 1) the user session identifier (obtained by means of the heuristic *HSession*), and 2) the use case executed in a user session (obtained by joining the user session identifier to the use case identifier generated by the heuristic *HUseCase*).

VI. Identification and analysis of deviations

The normative model – generated via Model to Model (M2M) transformation from the UML specification of the system – is a formal representation of the system scenarios, while the event logs – that result from the pre-processing of data logs – consist of sets of execution traces, where each trace describes a possible concrete behavior. The normative model is not exhaustive, since it considers a (set of) system behavior scenarios. Similarly, the event logs provide information about the running system, however they do not include a full description of the system behavior.

In this step of the approach, we consider two types of process mining techniques, implemented in ProM [13], to identify and analyze deviations with respect to the known behavior. Such deviations can reveal either attack patterns or correct behaviors which were not included in the initial UML specification. Firstly, we used the trace driven simulation [14] to identify deviations (subsection VI-A). Then, we applied the fuzzy model discovery technique [15] to analyze in depth such deviations in order to detect attacks and discover new usage scenarios (subsection VI-B).

We considered the logs of the SID BiD system, collected during the month of April, 2015, and the Petri Net model of Figure 4. The logs have been pre-processed to assign the user session as case-id (i.e., *HSession*), and they include a total of 2423 traces and 4805 different event types. More experiments on the case study, which also consider larger observation periods and the use case heuristics detailed in Section V, can be found in [16].

A. Identification of deviations

The trace driven simulation [14] replays the traces in the event log on the normative model, provided that a mapping is set between the transitions in the model and events in the logs. In the running example, the labeled transitions in Figure 4 have a correspondence with the events in the log, whereas the rest of transitions are assumed not observable. One of the simulation results is the fitness value, that is an indicator of the alignment of the traces with respect to the normative model: in the SID BiD system the fitness is about 58%, meaning that 42% of the traces in the log cannot be completely reproduced by the firing of a transition sequence, from the initial marking (i.e., place *Start*) to the final marking (i.e., place *End*) of the normative Petri Net model.

The ProM tool enables the visualization of the log-model alignments that indicate, for each type of trace, the sequence of event occurrences characterized by different colors depending on the type of alignment with the model. In particular, we are interested in the HTTP requests to resources that are not represented in the normative model since they reveal new behaviors: e.g., in the SID BiD logs there are two traces characterized by 14 HTTP requests related to *uploadify.swf* (cf. one of them in Figure 7). Similarly, four traces of different lengths (i.e., including between 6 and 13 HTTP requests) but all related to *FCKEditor*[6], a full featured

[6]URL: http://ckeditor.com/

GUI editor used by many web sites especially blog systems like WordPress, have been identified.

Besides, there are also five traces corresponding to the *Download article* use case scenario (i.e., *attachedFile* HTTP request) that are characterized by a very high number of *attachedFile* HTTP requests with respect to the rest of the traces in the log. In particular, the longest one includes 123 *attachedFile* events. Although such traces are fully aligned with the normative model, they deserve further analysis due to the high frequency of the event occurrences. Such traces have been identified manually, by observing the log-model alignments visualization produced by the ProM tool and by considering the third quartile as a frequency threshold. Therefore, the numbers of HTTP requests that fall above such a threshold are ouliers, and the execution traces including such requests are labeled as suspicious.

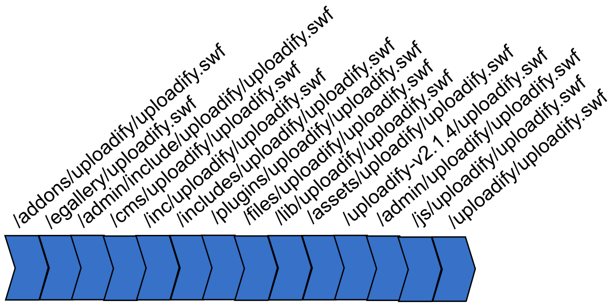

/addons/uploadify/uploadify.swf
/egallery/uploadify.swf
/admin/include/uploadify/uploadify.swf
/cms/uploadify/uploadify.swf
/inc/uploadify/uploadify.swf
/includes/uploadify/uploadify.swf
/plugins/uploadify/uploadify.swf
/files/uploadify/uploadify.swf
/lib/uploadify/uploadify.swf
/assets/uploadify/uploadify.swf
/uploadify-v2.1.4/uploadify.swf
/admin/uploadify/uploadify.swf
/js/uploadify/uploadify.swf
/uploadify/uploadify.swf

Figure 7. Anomalous trace.

B. Attack detection and process enhancement

The deviations from the expected behavior were further analyzed by filtering the logs according to the value of an event attribute, concretely the HTTP status code, and applying fuzzy mining discovery techniques [15]. The log filtering enables to select those traces that include HTTP requests characterized by the provided HTTP status code. Thus, the selected traces only include such HTTP requests, i.e., the other HTTP requests in the trace with a different HTTP status code are removed. The fuzzy models generated from the filtered logs are directed graphs, where each node represents an HTTP request and an arc between two nodes represents the causal relation between the HTTP requests. Observe that, due to the considered filtering criterion, the causal relation between two HTTP requests may be direct or indirect. Additionally, the fuzzy models incorporate several significance and correlation metrics useful to detect attack patterns (e.g., the frequency associated to a node/arc and the time proximity of two nodes).

The anomalous behaviors, that possibly represent attack patterns, are detected by considering HTTP request errors (i.e., 4XX and 5XX codes related to the client and server side, respectively) and correct HTTP requests with partially delivered resources (i.e., 206 code). Whereas, new behaviors can be discovered by considering successful HTTP requests (i.e., 200 code).

In the SID BiD system, different attack patterns can be discovered by using log filtering and fuzzy mining. For example, if we consider the HTTP status code 404 (i.e., not found resource), the fuzzy model generated from the filtered logs under analysis consists of a set of separate sub-models where different patterns are detected. Two of them are attempts to find vulnerabilities associated to the *uploadify.swf* script[7] and to the *FCKEditor*, respectively, as mentioned in the Subsection VI-A. Indeed the high proximity values of the nodes belonging to each sub-model (i.e., equal to one, the maximum value) indicate that the corresponding HTTP requests have the same timestamp, therefore we consider that they are automated.

On the other hand, the fuzzy model that was generated from the logs filtered by the HTTP status code 206 (Figure 8) represents iterative file download HTTP requests: similarly, the proximity value associated to the node with itself is one. Therefore, the high frequency of such requests within the same user session (i.e., a trace) and their simultaneous occurrences (i.e., same timestamp) reveal Denial of Service attack attempts.

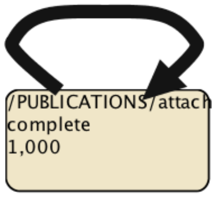


Figure 8. Fuzzy model from filtered logs (partial delivery).

In the experimental activities that were carried out with the SID BiD system [16], other interesting attack patterns were identified and detected, by following this analysis approach, such as: cross-site scripting (XSS) attempts, through the exploitation of the already mentioned vulnerabilities of the *uploadify* script, brute force attacks to get access passwords and attempts to edit publications by unauthorized users.

Finally, the fuzzy model discovered from the logs, which have been filtered by the HTTP status code 200 (i.e., successful request), reveals new behavioral patterns. The model includes four separated sub-models with a total of 10 legal new HTTP requests that are not represented in the initial UML specifications. These requests either refine the use case scenarios, initially modeled with the activity diagram, or belong to other use cases. In particular, one HTTP request is common to the use case scenarios already

[7]The database of the National Institute of Standards and Technology provides a list of known vulnerabilities associated to this script (see URL: https://web.nvd.nist.gov/view/vuln/search)

considered in the activity diagram of Figure 3, three HTTP requests are related to the *edit reference* use case and an HTTP request belongs to the *generate BibTex* use case (cf. Figure 2), whereas the rest of HTTP requests reveals new use cases (e.g., the help functionality).

VII. Conclusion

In this work, a method based on Process Mining and Model Driven Engineering has been proposed in order to improve the security of Web information systems. Besides, the proposed method has been applied to the SID BiD case study, and some promising preliminary results have been obtained. Despite the fact that the experimental results have been obtained by considering a particular technology to build Web information systems (in particular, JEE), the proposed approach is technology independent.

Currently, the analysis has been performed off-line and several different tools have been used as a support of the method, that is, the logs of the SID BiD system in the operational environment have been copied in order to perform the analysis in detail in a development environment. Nevertheless, we would like to fully automatize the proposed method to be used in DevOps environments in real time. In particular, our first next goal is incorporating a Security Information and Event Management system (SIEM), such as Zabbix[8], to monitor the activity of the SID BiD system. After that, we would like to automatically incorporate rules to the SIEM and to the Tomcat Web server in order to improve the security levels of the SID BiD system. Those rules must remove or reduce the conditions in which the attack patterns or threats (discovered with the process mining techniques) can happen. Finally, to validate the proposed method, we plan to apply it to other Web information systems in operation.

Acknowledgment

This work has been partially funded by the following projects: "Desarrollo de técnicas de detección de ciber-ataques en sistemas de información mediante minería de procesos" (UZCUD2016-TEC-06), "Infraestructuras críticas resistentes a ciber-ataques: Aplicando la minería de procesos y el diseño software orientado a la seguridad" [CyCriSec-TIN2014-58457-R], and "Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements" [DICE-H2020-644869].

References

[1] Symantec, "Internet Security Threat Report," Symantec Corporation, Tech. Rep., April 2016.

[2] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[3] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, 1st ed. Morgan & Claypool Publishers, 2012.

[4] J. Andress, *The Basics of Information Security: Understnding the Fundamentals of InfoSec in Theory and Practice*. Syngress, Boston, 2014.

[5] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*. Taylor & Francis Group, 2011.

[6] R. Accorsi, T. Stocker, and G. Müller, "On the exploitation of process mining for security audits: the process discovery case," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, 2013, pp. 1462–1468.

[7] F. Bezerra, J. Wainer, and W. M. P. van der Aalst, *Anomaly Detection Using Process Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 149–161.

[8] OMG, "Unified Modeling Language," Object Management Group, Tech. Rep., March 2015.

[9] J. López-Grao, J. Merseguer, and J. Campos, "From UML activity diagrams to Stochastic Petri nets: application to software performance engineering," in *Proceedings of the Fourth International Workshop on Software and Performance, WOSP 2004, Redwood Shores, California, USA, January 14-16, 2004*, 2004, pp. 25–36.

[10] S. Bernardi, J. Campos, and J. Merseguer, "Timing-Failure Risk Assessment of UML Design Using Time Petri Net Bound Techniques," *IEEE Trans. Industrial Informatics*, vol. 7, no. 1, pp. 90–104, 2011.

[11] A. Gómez, C. Joubert, and J. Merseguer, "A Tool for Assessing Performance Requirements of Data-Intensive Applications," in *Proc. of the XXIV National Conference of Concurrency and Distributed Systems (JCDS 2016)*, 2016, pp. 159–169.

[12] ISO, "Systems and software engineering – High-level Petri nets – Part 2: Transfer format," ISO/IEC, Geneva, Switzerland, Tech. Rep. 15909-2:2011, 2008.

[13] B. F. Van Dongen *et al.*, "The ProM framework: A new era in process mining tool support," in *Applications and Theory of Petri Nets 2005*. Springer, 2005, pp. 444–454.

[14] W. Van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 182–192, 2012.

[15] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining - adaptive process simplification based on multi-perspective metrics," in *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, ser. Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann, Eds., vol. 4714. Springer, 2007, pp. 328–343.

[16] R. Piracés-Alastuey, "PMS methodology," URL: http://sid.cps.unizar.es/PMS/, in Spanish.

[8]Zabbix: http://www.zabbix.com/