

A Self-protecting Approach for Service-oriented Mobile Applications

Ronaldo Rodrigues Martins, Marcos Paulo de Oliveira Camargo, William Filisbino Passini,
Gabriel Nagassaki Campos and Frank José Affonso^a

*Department of Statistics, Applied Mathematics and Computation, São Paulo State University – UNESP,
PO Box 178, Rio Claro, São Paulo, 13506-900, Brazil*

Keywords: Self-protecting, Mobile Applications, Web Service, Security.

Abstract: The evolution of software systems in the last 10 years has brought new challenges for the development area, especially for service-oriented Mobile Applications (MobApps). In the mobile computing domain, the integration of MobApps into service-based systems has been a feasible alternative to boost the capacity of processing and storage of such applications. In parallel, this type of application needs monitoring approaches mainly due to the need of dealing with a large number of users, continuous changes in the execution environment, and security threats. Besides that, most MobApps do not present the self-protecting property by default, resulting in a number of adverse situations, such as integrity of execution, reliability, security, and adaptations at runtime. The principal contribution of this paper is an approach based on MAPE-K (Monitor-Analyze-Plan-Execute over Knowledge) loop and machine learning techniques to ensure self-protecting features in MobApps, in particular, those based on services. Experimental results showed that this approach can autonomously and dynamically mitigate threats, making these applications more trustworthy and intrusion-safe. Our approach has good potential to contribute to the development of MobApps, going beyond existing approaches.


1 INTRODUCTION

Nowadays, our society has become increasingly dependent on software systems. In this scenario, it can be also noted that most human daily tasks are managed by Mobile Applications (MobApps) embedded into mobile or smart devices (e.g., smartphones, tablets, hybrid devices, smart-TVs, smart-watches, among others), which enable on-line access to information regardless of the users' location (Aghav and Sharma, 2011). These systems deal with complex structures that enable the interpretation of the context in which they are inserted and, at the same time, with extra requirements that have become them more versatile, extensible, resilient, dependable, robust, recoverable, customizable, configurable, and changeable (Salehie and Tahvildari, 2009).

The integration between MobApps and SOA-based (Service-Oriented Architecture) systems has been shown as an alternative to overcome limitations related to computation-intensive tasks, which can demand an excessive amount of battery power or storage space (Aghav and Sharma, 2011). Zahrani (2016)

argued that MobApps based on services can benefit from cloud platforms to perform distributed processing on multiple servers and access data remotely from different machines instead of using its own device. These applications need monitoring approaches to deal with large-scale environments to meet many clients and to mitigate adverse situations of security/threats (Sarker et al., 2020). Because of the constant changes unannounced in the execution environment, protect such applications through approaches based on the self-protecting property can be a feasible solution (Lara et al., 2019).

Based on the exposed scenario, there is a growing interest in both academia and industry in the development of solutions and theories to support the evolution of service-oriented MobApps that require self-protecting property. Self-protecting solutions must identify old and new attacks/threats through proactive and/or reactive strategies, i.e., they must anticipate the attacks/threats known in both academia and software industry, besides enabling the gradual identification and treatment of the new ones. Despite evident interest from both industry and academia, to our best knowledge, there is no self-protecting approach for MobApps that encompasses the afore-

^a  <https://orcid.org/0000-0002-5784-6248>

mentioned requirements. To do so, we conducted a Systematic Mapping Study (SMS) (Petersen et al., 2015) to identify secondary and primary studies that deal with MobApps that require the self-protecting property (Martins et al., 2021). Thus, an approach based on MAPE-K (Monitor-Analyze-Plan-Execute over Knowledge) loop (IBM, 2005) and machine learning techniques to ensure self-protecting features in MobApps is proposed in this paper (Sarker et al., 2020). This approach was designed based on the best practices of software engineering, the results of the aforementioned SMS, and the main security risks/threats (OWASP, 2021a; Martin et al., 2021).

The main contribution of this paper is a self-protecting approach to support the development of service-oriented MobApps. In short, this approach enables to address server-side security requirements, making the design of such applications easier and trustworthy. From the design viewpoint, our approach was built to deal with attacks and/or threats in both proactive and reactive strategies. To do so, it addresses the main security threats reported in the industry through learning techniques, which have automated the process of threats detection with excellent accuracy. Moreover, in order to expand the capacity to deal with old and new types of attacks/threats, our approach enables us to incorporate algorithms for the classification of vulnerabilities (i.e., proactive strategy), and recommendation of solutions (i.e., reactive strategy). Indirectly, our approach can benefit different research communities and practitioners, providing a solution that can guide future research and innovations focused on security strategies, implementations, frameworks, tools, among others. Thus, we believe to have created a favorable scenario of development, since our approach enables us to design service-oriented MobApps supported by self-protecting mechanisms that can make them more trustworthy and safe.

This paper is organized as follows. Section 2 presents the background and related work. A description of our approach is reported in Section 3. Section 4 presents a proof of concepts to show the applicability of our approach. Finally, Section 5 summarizes our conclusions and perspectives for further research.

2 BACKGROUND AND RELATED WORK

This section presents the background and related work that contributed to the development of our approach. Initially, concepts of self-protecting systems

are described. Next, related work on self-protecting solutions is addressed.

Self-protecting Systems. Adaptive security model or self-protecting mechanisms are synonyms for the capability of changing a system's security at runtime based on a certain level of threat. Security concerns refer to practices and processes that aim to ensure the confidentiality, availability, and integrity of data by restricting data usage or access by unauthorized entities. Tziakouris et al. (2018) defined "self-adaptive security systems as any solutions that can protect systems/users/data against runtime threats via the enforcement of alternate/adjustable defensive strategies". According to Chopra and Singh (2011), the principal purpose of self-protecting solutions is to defend the execution environment or applications against malicious intentional actions. Self-protecting systems should scan for suspicious activities and react to them without users knowing that such protection is in execution. Yuan and Malek (2012) defined self-protecting as an essential property for self-management of autonomic computing systems from two perspectives: (i) reactive, the system automatically defends against malicious attacks or cascading failures; and (ii) proactive, the system anticipates security problems and takes decisions to mitigate them.

As related work, Dey et al. (2015) developed a context-adaptive security framework for cloud-based MobApps that aims to provide an extra security layer, besides improving the Quality of Service (QoS) and reliability of such applications. This framework creates a secure session between MobApps and the cloud server, providing server security by checking patterns of incoming traffic based on a learning system.

A context-aware adaptive security framework was proposed by Mowafi et al. (2014), which provides a multi-security incubator so that it executes a MobApp. They developed this framework based on policy selection, decision making, adaptive learning, and recommendation and feedback systems. By running each application in its own incubator as a standalone application, it is possible to run security and communication mechanisms within the incubator and optimize the security of such applications.

Amoud and Roudies (2017) proposed a self-adaptive security approach for mobile devices based on the MAPE-K loop. In short, this approach enables the dynamic negotiation of security policies and automatic reconfiguration of security levels. Thus, new security policies can be instantiated at runtime so that they meet the new security needs during changes that can occur in the execution environment.

According to Zahrani (2016), it is common to see data storage and replication in different locations

as a way of overcoming data loss and availability problems (e.g., cloud computing). The author proposed a self-protecting mechanism for Mobile Cloud Computing (MCC) based on MAPE-K loop, which aims to enable the self-protecting capability for this application type (MCC) when data is transferred from the mobile device to cloud.

As stated in Saxena et al. (2007), security is commonly treated as a system's static component. This principle cannot be applied to systems that are developed to operate in different environments, since they deal with devices connected in divergent contexts. Thus, it is necessary to develop solutions able to monitor every system's aspect and intelligence to carry out modifications/adaptations whenever necessary. In a real scenario, it is not always possible to predict all attack scenarios during the life cycle of the system. In this sense, implementing different policies for each environment solution can optimize the security levels of a system.

3 SELF-PROTECTING APPROACH

As our approach aims to address service-oriented MobApps that require the self-protecting property, we have gathered the main OWASP security risks for three types of applications (i.e., web, mobile, and services). These risks are based on data collected and in discussions with the software development community, classifying the risks according to OWASP (2021c). We also focused our concerns in the main OWASP threats (top 10) (OWASP, 2021a) and complemented with the Common Weakness Enumeration (top 25) (Martin et al., 2021) because of their prevalence on the Internet and their relevance to the application domain of our proposal. However, it is noteworthy the approach proposed in this paper can treat threats not contained in aforementioned lists.

Figure 1 shows a detailed view of our self-protecting approach. In short, it received the main contributions of an SMS (Martins et al., 2021), the studies reported in Section 2, and the aforementioned security risks. In this sense, we highlight the MAPE-K control loop (IBM, 2005), the learning techniques, and the modular architectural organization (Gamma et al., 1995), which enables our approach to be flexible when scaled to meet new threats and/or vulnerabilities types (Sarker et al., 2020). From an operational viewpoint, the proposed approach aims to address security threats reported in this section through proactive and reactive strategies. The first aims to identify the known attacks/threats that may occur in

an application at runtime. The purpose of this identification is to anticipate problems such as QoS degradation, interruption of services, improper access to data, among others. The second enables to deal with the aforementioned problems so that applications are not compromised, both from an operational viewpoint and from undue exposure of information. Finally, it is worth mentioning that we adopted an architectural organization based on the decentralized topology to protect the self-protecting module (i.e., the "protector" of our approach) (Yuan et al., 2014).

As can be observed in Figure 1, our approach contemplates the development of service-oriented MobApps in both phases: design and runtime. Another important aspect to be highlighted in this approach is its security scope, since it was designed to operate in the application layer (server-side), where occurs communication between a MobApp and the Web services used by it. Regarding the services, this approach enables the development of MobApps based on SOAP and/or RESTful that can be organized through orchestration, choreography, or simple services. To do so, this approach was organized in four activities, namely: (1) Development; (2) Learning; (3) Execution; and (4) Monitoring. Next, details of each activity is addressed.

Activity 1 represents the development of service-oriented MobApps, which should be conducted based on security guidelines and supported by an automated process for identifying vulnerabilities in the design phase OWASP (2021c). As our approach uses learning algorithms to classify (i.e., predict) the risks/weaknesses of an application, it is necessary to define the "monitoring points" so that it can collect and analyze that data at runtime (Activity 2). We consider this step fundamental to our approach because a good definition of the data can significantly improve the classification and/or prediction of a learning-based system (Kamath and Choppella, 2017). Next, developers can evaluate the vulnerabilities of the application before inserting it into the execution environment through automated processes, which can provide parameters regarding security weaknesses, possibilities of invasion, among others. Based on this context, by evaluating self-protecting as a broad, comprehensive, dynamic, and evolutionary issue, we would like to emphasize that the approach proposed in this paper is not a "silver bullet" for all threat types for the service-oriented MobApps. We recommend developers to use a list of security techniques developed by OWASP Proactive Controls (OWASP, 2021b) during the development activity, since this list provides ten important items that can optimize threat mitigation. Thus, in

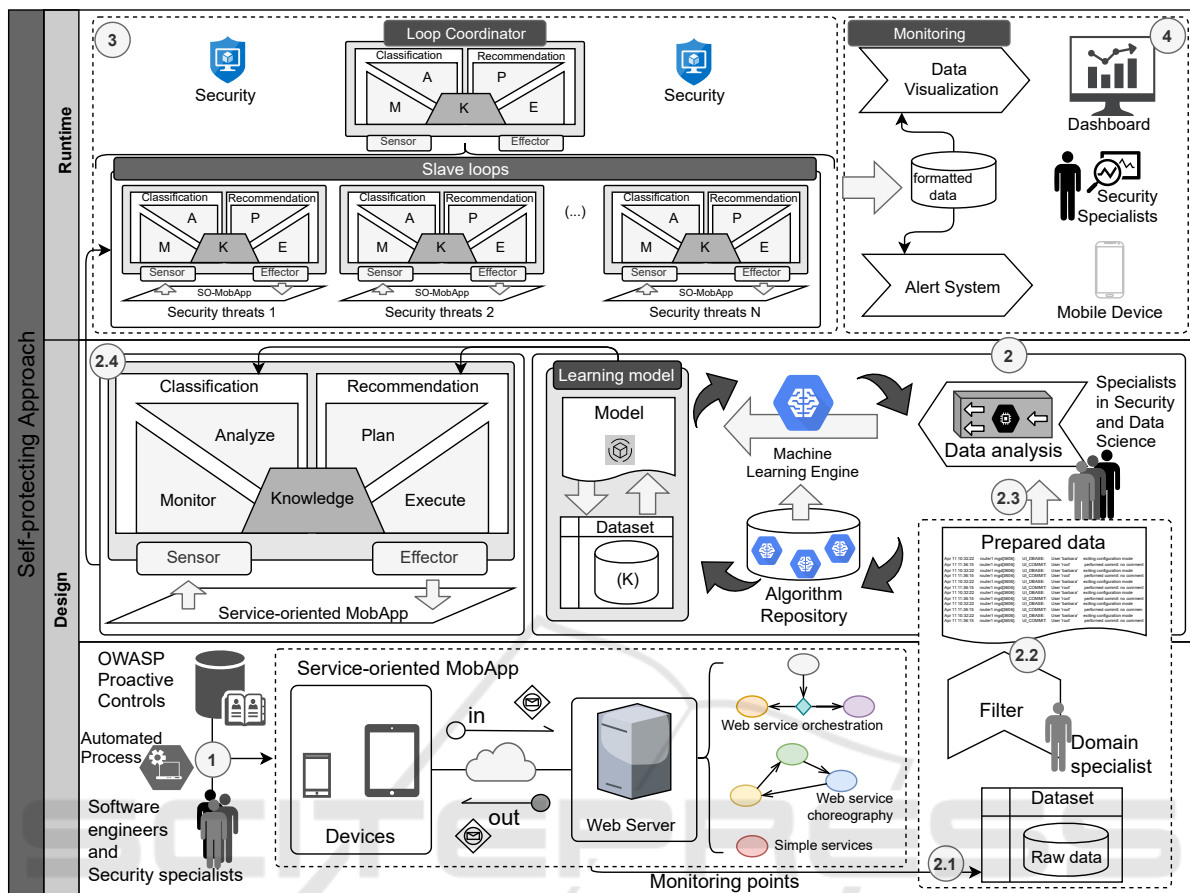


Figure 1: Self-Protecting approach.

order to guide the security-oriented development, two considerations regarding the major risks can be highlighted. The first is related to the use of an additional layer to the system for mitigating some security risks. For instance, the developers must implement data validation on the front-end and back-end sides so that threats such as code injection and cross-site scripting can be avoided. In parallel, they can implement a data encryption layer to avoid exposing data and metadata. The second is related to code security. As our approach aims to deal with security threats at runtime, services can be modified or replaced at runtime without the perception of their stakeholders. Thus, some threats are implicitly circumvented, namely: integration of data and services, client code quality, code tampering, exposure of binary code, among others.

Activity 2 represents the development process of the learning models that will be used to identify vulnerabilities and attacks/threats at runtime. This activity occurs in incremental and iterative cycles with Activity 1, since it is necessary to define and refine the data that will be collected from the service-oriented MobApps during their execution and which learning

algorithms will be used according to the vulnerabilities and attacks/threats that are intended to be addressed. To do so, we organize this activity as follows: (i) Step 2.1 specifies the problem to be treated and the organization of the dataset (i.e., data from third parties and data collected with the preliminary execution of the system) that will be used to train the model so that an anomaly (i.e., vulnerability or attack) in service-oriented MobApps can be identified; (ii) Step 2.2 represents the data preparation task that should be conducted for the elimination of undesired data (“Filter” phase). Next, it conducts the selection of features (“Prepared data” Phase) so that the dataset is adequate to the problem that is being addressed; (iii) Step 2.3 selects the algorithms for classification and recommendation modules, which represent the learning model of our approach (Step 2.4) for each vulnerability, attack, or anomaly; and (iv) Step 2.4 represents the final learning models that will be deployed in the execution environment, which are referenced from this point forward as supervisor systems.

In relation to the presented steps, the calibration of the learning models is a task required for our ap-

proach in order to evaluate the identification accuracy of vulnerabilities, attacks/threats, or anomalies according to the labeled data (i.e., that one provided by the security specialist in relation to the data acquired from the execution of the system). Moreover, it is noteworthy that other algorithms can be coupled to our approach without additional implementation because it was designed to be flexible and scalable (i.e., a common interface is available for the learning algorithms) (Gamma et al., 1995). According to Psaiar and Dustdar (2011), this process can optimize the algorithm performance for both modules. Another important aspect that should be taken into consideration when choosing machine learning algorithms is the learning type, which can be classified in three categories: (i) **supervised**: when the future can be predicted based on data learned in the past. The training dataset is used to produce an inferred function to make future predictions. After that, the system can classify new inputs from labeled data; (ii) **unsupervised**: when the information is used to train is neither marked nor classified. In this type of learning, the role of the machine is to group unsorted information according to patterns, similarities, and differences with no prior training data; and (iii) **semi-supervised**: this type of learning can be considered as an intermediate model because it uses both labeled and unlabeled data for training. In short, we can use this type in scenarios where it is difficult to produce an accurate model, therefore, semi-supervised techniques can increase the size of the training data.

Regarding the operating mode (Activity 3), our approach acts as a non-intrusive supervision modality, i.e., a supervisor system is responsible for monitoring the internal states of a service-oriented MobApp. As can be observed in Figure 1 (Activity 3), there is a control loop coordinator and multiple slave loops. Each slave loop deals with a type of vulnerability, attack, threat, or anomaly, since each security problem requires specific features and behaviors that should be considered in the execution of both modules (i.e., classification and recommendation). In short, sensors are responsible for capturing parameters from the execution environment for the supervisor system. Next, the classification module classifies these parameters to identify the changes occurred in each monitoring point. Based on this classification and the data collected from the environment, an adaptation plan is prepared by the recommendation module to establish a solution for the identified risk/threats. Before it becomes an effective solution, such recommendation must be tested in order to ensure that no “collateral effects” will be propagated to the application (i.e., service-oriented MobApp). Effectors deal with

the “selected solution” after its testing activities are performed, applying it to the application. When a vulnerability, attack, threat or anomaly is identified, the learning models are updated, characterizing an incremental and dynamic learning strategy. Finally, in order to guarantee that the solution proposed does not become unfeasible, we recommend that the supervisor systems (i.e., coordinator and slaves) be deployed in a different host than the application that is being monitored in order to protect the protector of our approach. Moreover, we recommend that security guidelines are applied, as suggested by (OWASP, 2021a) and (Martin et al., 2021).

Finally, as suggested by the OWASP AppSensor (Watson et al., 2015), monitoring an application (Activity 4) through automated processes helps in mitigating/identifying security threats. In this sense, we have elaborated a monitoring environment (i.e., a dashboard with parameters related to the monitoring points) together with a message notification system. Essentially, these systems receive the classification module data containing the information collected, and the anomalies identified. In short, we designed these systems to assist the security specialists, providing information about attacks and vulnerabilities of an application in a quick and objective way.

4 PROOF OF CONCEPT

This section presents a proof of concept conducted to evaluate the applicability of our approach. As a subject application for our empirical analysis, we have selected a public dataset named CSIC TORPEDA 2012, which provides 74,133 labeled HTTP requests to an e-commerce web application in XML (eXtensible Markup Language) format (Torrano et al., 2012). According to these authors, this dataset is composed of 8,363 normal requests, 16,459 anomalous requests, and 49,311 malicious requests. They injected eight types of threats in the last type of request to simulate attacks in a web application, namely: 43,013 SQLi and variants, 4,818 XSS and variants, 412 Buffer Overflow, 41 Format String, 74 LDAPi, 451 SSI, 175 XPath, and 327 CRLF. Based on the exposed context, for reasons of scope and space, this paper will only address scenarios for the identification/classification of such threats and notification of the security specialist in relation to threat types identified in these scenarios. Finally, as the data in this dataset is labeled, our approach will be instantiated to work with supervised learning models.

Figure 2 presents an overview of the pre-processing step of our approach, which represents

the preparation of the raw data (e.g., Step 2.1 of our approach – Figure 1). To do so, we developed a Java application called XML2CSV, which enabled us to convert the log files in XML to a Weka input format (e.g., CSV – Comma-Separated Values). Then, using Weka’s analysis process, we selected five features to classify requests as normal or anomalous, namely: lengthRequest, lengthArguments, numberArguments, lengthPath, and numberSpecialCharsPath. After this step, we choose six algorithms to perform the classification of the requests. The description of these activities represents the execution of Steps 2.2 and 2.3 of our approach (see Figure 1). Finally, we performed the model calibration to detect anomalous requests and obtained the following accuracy distribution for the selected algorithms: Logistic 95.85%, LibSVM 99.16%, AdaBoostM1 98.24%, J48 99.61%, K-Nearest Neighbor 99.89%, and Random Forest 99.89%. A brief description of these algorithms is presented in Table 2, which shows the classification accuracy of the attacks.

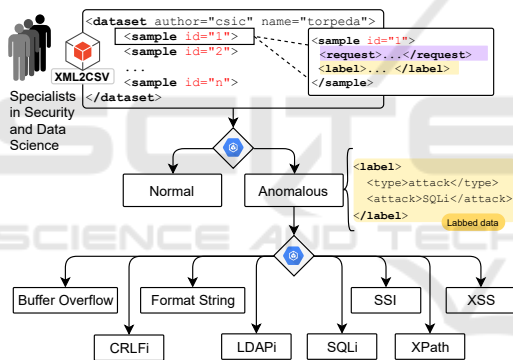


Figure 2: Pre-Processing process of our approach.

After identifying the requests as anomalous, the next step is to classify them as one type of threat, as illustrated in Figure 2, which represents the Step 2.3 of our approach. To show how the operation must be performed, the type of SQLi attack was selected to be presented in more detail. Initially, the CSV file must be loaded in the Weka tool for applying three filters: (i) RemoveDuplicates, which enables to remove all duplicate instances; (ii) Randomize, which randomly shuffles the order of instances; and (iii) ReplaceMissingWithUserConstant, which enables to replace all missing values for nominal, string, numeric, and date attributes in the dataset. Next, we generated an ARFF file containing the specification and data (i.e., learning model) for this type of attack. Regarding the specification, we used a set of 58 features defined by Bhagwani et al. (2019), as shown in Table 1.

Table 1: SQLi attack features (Bhagwani et al., 2019).

'--', '/**/', '%', '+', 'r', ';', '#', '=', '[', ']', '(', ')', '^', '*', 'char', '\', '-', '<', '>', '.', ' ', '<>', '<=', '>=', '&&', ' ', ':', '!=', 'count', 'into', 'or', 'and', 'not', 'null', 'select', 'union', 'insert', 'update', 'delete', 'drop', 'replace', 'all', 'any', 'from', 'user', 'where', 'sp', 'xp', 'like', 'exec', 'admin', 'table', 'sleep', 'commit', '()', 'between',

To complete Activity 2 of our approach, we must conduct the model evaluation process so that Step 2.3 can be consolidated. Thus, it can be said that we have an initial dataset (i.e., knowledge) and a learning model that can be deployed in the classification module to deal with SQLi attacks at runtime (see Step 2.4 – Figure 1). Regarding the model evaluation, it is noteworthy that we analyzed the classification for this type of attack using the aforementioned six algorithms, as shown in Table 2. Our goal in presenting these results is not to provide qualitative and/or comparative evidence regarding the use of such algorithms, since security and data science specialists can choose algorithms according to the needs of the application and results of this calibration process.

Analyzing the data in the Table 2, J48 algorithm showed the best result for the CRLFfi, SQLi, and Xpath attacks, with an accuracy of 84.85%, 98.67%, and 99.45 respectively. 98.96% was the accuracy achieved by the LibSVM, AdaBoostM1, and J48 algorithms for the LDAPi attack. Similarly, 95.45% was the accuracy achieved by the Logistic and J48 algorithms for the SSI attack. Finally, K-Nearest Neighbor algorithm showed the highest accuracy of 99.53% for the XSS attack.

Figure 3 illustrates how the slave loops are used by the coordinator loop. As can be observed by the area highlighted in light gray (solid border), which represents a slave loop for the CRLFfi attack, new data is collected so that this type of attack can be identified. We represent the new requests by tuples composed of information for each attack type (i.e., x, y, z, w, ..., empty), which, after processing activity of each loop, can be classified as attacks (i.e., red square) or normal request (i.e., blue square). The coordinator loop monitors the slave loops so that the classified data is collected. Then, this loop gathers the data and notifies the security specialist about which attacks the application is suffering so that he can take assertive decisions, avoiding that the application is compromised.

Table 2: Attack detection in %.

Algorithms	CRLF _i	LDAP _i	SQL _i	SSI	Xpath	XSS
Logistic	80.30	98.93	90.48	95.45	99.18	95.08
LibSVM	81.82	98.96	95.93	87.34	99.15	96.80
AdaBoostM1	83.33	98.96	91.89	95.13	99.15	93.04
J48	84.85	98.96	98.67	95.45	99.45	99.26
K-Nearest Neighbor	72.73	97.95	98.50	92.21	99.08	99.53
Random Forest	74.24	97.99	98.60	91.23	99.10	99.49

Where *Logistic* is a regression algorithm for binary classification, *LibSVM* is a library for Support Vector Machines, *AdaBoostM1* is a classifier for nominal class problems, *J48* is a classifier based on tree, and *K-Nearest Neighbor* and *Random Forest* are algorithms to deal with classification and regression problems.

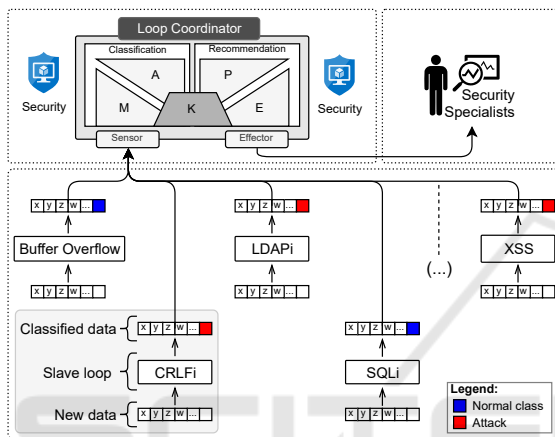


Figure 3: Classification process.

5 CONCLUSIONS

A self-protecting approach to support the development of service-oriented MobApps was presented in this paper. In short, our approach uses machine learning techniques and MAPE-K loop to deal with different types of attacks/threats. The design of our approach is based on results of an SMS conducted by Martins et al. (2021) and OWASP security risks (Foundation, 2021; Martin et al., 2021). On the other hand, we would like to emphasize that our approach is not a “silver bullet” for all threat types. Based on the results of this paper, we are providing a solution capable of guiding developers and organizations interested in the development of this type of application in order to optimize the threat mitigation and the solution customization faced during the development. Thus, we believe that our approach can boost the development of industrial solutions and future research in these areas. The main contributions of this paper are: (i) self-protecting approach that can facilitate the development of this application type, benefiting several research communities, namely: Service Computing, Mobile Computing, and

Security; and (ii) stakeholders interested in how the self-protecting property has been used for the development of service-oriented MobApps to make such applications more trustworthy and intrusion-safe.

Regarding future work, at least two activities are intended: (i) conduction of more case studies or proof of concepts to evaluate our approach, including different machine learning algorithms, learning approaches, and security scenarios; and (ii) use of this approach in a larger real environment of development and execution. Therefore, based on the content presented in this paper, a positive research scenario can be idealized, enabling this approach to become an effective contribution to the involved communities.

ACKNOWLEDGEMENTS

This research is supported by UNESP’s Pro-Rectorate of Research (PROPe/UNESP), the São Paulo Research Foundation (FAPESP) - Brazil (Grant: 2019/21510-3), and the LINEAS and RADIAR Project (UNESP/Petrobras/FUNDUNESP Cooperation) - Brazil (Grants: 2017/00502-7, 5850.0102453.16.9).

REFERENCES

Aghav, J. and Sharma, N. (2011). A software architecture for provisioning of mobile services: An OSGi implementation. In *The 7th International Conference on Perspective Technologies and Methods in MEMS Design*, pages 24–27.

Amoud, M. and Roudies, O. (2017). Dynamic adaptation and reconfiguration of security in mobile devices. In *2017 International Conference On Cyber Incident Response, Coordination, Containment Control (Cyber Incident)*, pages 1–6.

Bhagwani, H., Negi, R., Dutta, A. K., Handa, A., Kumar, N., and Shukla, S. K. (2019). Automated classification of web-application attacks for intrusion detection. In Bhasin, S., Mendelson, A., and Nandi, M.,

- editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 123–141, Cham. Springer International Publishing.
- Chopra, I. and Singh, M. (2011). Sasm-an approach towards self-protection in grid computing. In Dua, S., Sahni, S., and Goyal, D. P., editors, *Information Intelligence, Systems, Technology and Management*, Communications in Computer and Information Science, pages 149–159. Springer Berlin Heidelberg.
- Dey, S., Sampalli, S., and Ye, Q. (2015). A context-adaptive security framework for mobile cloud computing. In *11th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 89–95.
- Foundation, O. (2021). Owasp mobile security project. [On-line]. Available: <http://tiny.cc/owasp-top10-msp>, Accessed on February 26, 2021,.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- IBM (2005). An architectural blueprint for autonomic computing. [On-line]. Available: <http://tiny.cc/ibm-loop>, Third Edition, Accessed on February 21, 2021.
- Kamath, U. and Choppella, K. (2017). *Mastering Java Machine Learning: A Java Developer's Guide to Implementing Machine Learning and Big Data Architectures*. Packt Publishing.
- Lara, E., Aguilar, L., Sanchez, M. A., and García, J. A. (2019). Adaptive security based on mape-k: A survey. In Sanchez, M. A., Aguilar, L., Castañón-Puga, M., and Rodríguez, A., editors, *Applied Decision-Making: Applications in Computer Sciences and Engineering*, pages 157–183, Cham. Springer International Publishing.
- Martin, B., Brown, M., Paller, A., and Kirby, D. (2021). 2011 cwe/sans top 25 most dangerous software errors. [On-line]. Available: <https://cwe.mitre.org/top25/index.html>, Accessed on February 26, 2021,.
- Martins, R. R., de Oliveira Camargo, M. P., Passini, W. F., Campos, G. N., and Affonso, F. J. (2021). Mapping study on self-protecting property for service-based mobile applications domain. [On-line]. Available: <https://drive.google.com/file/d/1yxQ58VrJ06Ks5T3h7aXk9dAi7bm3wUqO/view>, Accessed on February 26, 2021.
- Mowafi, Y., Abou-Tair, D., Aqarbeh, T., Abilov, M., Dmitriyev, V., and Gomez, J. M. (2014). A context-aware adaptive security framework for mobile applications. In *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, ICCASA '14, pages 147–153, Brussels, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- OWASP (2021a). The owasp foundation. [On-line]. Available: <https://www.owasp.org>, Accessed on February 26, 2021,.
- OWASP (2021b). Owasp proactive controls. [On-line]. Available: <https://owasp.org/www-project-proactive-controls>, Accessed on February 26, 2021,.
- OWASP (2021c). Owasp risk rating methodology. [On-line]. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology, Accessed on February 26, 2021,.
- Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18.
- Psaier, H. and Dustdar, S. (2011). A survey on self-healing systems: approaches and systems. *Computing*, 91(1):43–73.
- Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4:1–42.
- Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., and Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. *Journal of Big Data*, 7(1):41.
- Saxena, A., Lacoste, M., Jarboui, T., Lücking, U., and Steinke, B. (2007). A software framework for autonomic security in pervasive environments. In McDaniel, P. and Gupta, S. K., editors, *Information Systems Security*, pages 91–109, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Torrano, C., Pérez, A., and Álvarez, G. (2012). Http csic torpeda 2012 dataset. [On-line]. Available: <https://www.tic.itefi.csic.es/torpeda/datasets.html>, Accessed on February 26, 2021.
- Tziakouris, G., Bahsoon, R., and Babar, M. A. (2018). A survey on self-adaptive security for large-scale open environments. *ACM Computing Surveys*, 51(5):100:1–100:42.
- Watson, C., Groves, D., and Melton, J. (2015). Appsensor guide - application-specific real time attack detection & response. version 2.0. [On-line]. Published 27th July 2015. Available: <http://tiny.cc/owasp-appsensor>, Accessed on February 26, 2021,.
- Yuan, E., Esfahani, N., and Malek, S. (2014). A systematic survey of self-protecting software systems. *ACM Transactions on Autonomous and Adaptive Systems*, 8(4).
- Yuan, E. and Malek, S. (2012). A taxonomy and survey of self-protecting software systems. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 109–118.
- Zahrani, M. (2016). Self-protection and security in mobile cloud computing. *Research Journal of Information Technology*, 8(1-2):47–54.