Automated Interpretation and Integration of Security Tools Using Semantic Knowledge

Chadni Islam[1,2,3(✉)], M. Ali Babar[1,2(✉)], and Surya Nepal[3(✉)]

[1] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
{chadni.islam,ali.babar}@adelaide.edu.au
[2] CREST Centre, Adelaide, SA 5005, Australia
[3] Data61, CSIRO, Sydney, NSW 2122, Australia
surya.nepal@data6l.csiro.au
http://crest-center.net

Abstract. A security orchestration platform aims at integrating the activities performed by multi-vendor security tools to streamline the required incident response process. To make such a platform useful in practice in a Security Operation Center (SOC), we need to address three key challenges: interpretability, interoperability, and automation. In this paper, we proposed a novel semantic integration approach to automatically select and integrate security tools with essential capability for auto-execution of an incident response process in a security orchestration platform. The capability of security tools and the activities of the incident response process are formalized using ontologies, which have been used for NLP based approach to classify the activities for the emerging incident response processes. The developed ontologies and NLP approaches have been used for an interoperability model for selection and integration of security tools at runtime for the successful execution of an incident response process. Experimental results demonstrate the feasibility of the classifier and interoperability model for achieving interpretability, interoperability, and automation of security tools integrated into a security orchestration platform.

Keywords: Security orchestration · Ontological model · Self-adaptive · Automation and interoperability · Security automation

1 Introduction

The Security Operation Center (SOC) of an organization uses a variety of security tools, developed by different vendors, to protect an organization's Information and Communication Technology (ICT) infrastructure and Business Application (BA) [1–3]. Examples of such tools are Intrusion Detection System (IDS), Firewall, Endpoint Detection and Response (EDR), and Security Information and Event Management (SIEM). According to a recent report by Enterprise Strategy Group [4], on average a SOC has 25 different security tools, and this number goes up to 100 for some SOCs. Most of these tools work independently. The security experts of a SOC are expected to monitor and analyze the activities (i.e., validate alerts, correlate log, and remove malware) of these security tools to respond to an incident [1, 5–7]. The continuous

P. Giorgini and B. Weber (Eds.): CAiSE 2019, LNCS 11483, pp. 513–528, 2019.
https://doi.org/10.1007/978-3-030-21290-2_32

process of monitoring and analyzing the security activities are time-consuming, tedious and repetitive [5, 8].

Most SOC in recent years uses Security Orchestration Platform (SecOrP) to orchestrate the activities of security tools and automate the repetitive tasks manually performed by security experts [5–7]. Deployment of a SecOrP requires an organization to assess their existing security tools' capabilities (e.g., intrusion detection, log management, packet sniffing, and log correlation) and prepare an Incident Response Plan (IRP) [5–8]. An IRP is a sequence of activities that are performed by various security tools. Based on the assessment of an organization's existing tools and requirements, APIs or plugins are developed for integrating security tools into SecOrP and rules are defined to orchestration and automate the IRP [1, 5, 9].

Emerging threat behaviors and variations in organization's infrastructure cause experts to change the deployment and execution environment of SecOrP, such as the integration of new tools, updates of tools capability or modification of an IRP [3, 7, 10]. Existing SecOrPs, however, are not adaptive towards such changes [3, 7, 10]. Experts must sufficiently understand the APIs and rules of SecOrP to make it adaptive to the changes by defining new rules or developing new APIs [5, 11, 12]. Human intervention is required to adjust the changes because security tools are not *interoperable* and SecOrP cannot *interpret* security tools' activities and their input and generated data [12, 13]. A recent study by the SANS Institute *(Escal Institute of Advanced Technologies)* has revealed that the integration of security tools is the third most challenging task of SOC [14].

SecOrP requires the semantic knowledge to formalize various inputs, outputs, and activities of security tools. The formalized concepts enable a SecOrP to *interpret* the changes in runtime environment and *automate* the execution of modified or new IRP without any human intervention. Ontologies can be used to provide the required formal specification to support *interoperability* and *semantic integration* of security tools in a SecOrP without any human involvement [15, 16]. Semantic integration refers to the ability of SecOrP to understand the semantics of the input or output of security tools. A SecOrP can semantically interpret the activities of security tools when the formalization incorporates semantic integration of security tools.

The process of defining a suitable ontology is not straightforward [17]. A well-built ontology depends on domain expertise. Formalizing various security tools and the activities of IRP are challenging due to the ambiguity of the terminology used by different vendors. The features of security tools and activities are defined using Natural Language; same activity is defined using different terms in different IRP. The development of ontology is an incremental process. Domain experts require to perform manual tasks to keep ontologies updated as per the new knowledge.

We propose an integration framework for SecOrP that integrates the data generated by different security tools to automate the execution of an IRP by making security tools *interoperable*. The proposed *integration framework* consists of an *ontological model*, a *prediction module* and an *annotation module*. We have formalized the core concepts of SecOrP in an ontology that are required to automate the execution of an IRP. We have followed a systematic approach to define the classes of our ontology and the relationships among the classes.

We have designed and developed a prediction module utilizing the existing Natural Language Processing (NLP) and Machine Learning (ML) techniques to automatically classify the activities with text description according to the ontology. For a new activity description in an IRP, we have performed a text-based similarity measure with the existing list of activity description. We have defined a threshold for the similarity measure that is used to invoke the prediction module when the similarity score is above the threshold. For a similarity score below the threshold, we have designed an annotation module to generate and recommend the possible classes to experts and automatically annotate the new classes in ontology after an expert selects the classes.

We have designed and implemented an interoperability model to select the best suite of tools that have the required capability to execute an IRP. We check the compatibility of the set of selected tools for interoperability based on their capabilities in terms of their input, output and execution environment. In this paper, we do not show the development and evaluation of the ontology; instead, we demonstrate the use of the ontology by the prediction module and interoperability model for auto-execution of IRPs. Following are the key contributions of our work expected in this paper.

- An ontological model to formalize the diverse activities and capabilities of security tools (ref. Sect. 4.1).
- A prediction module to automatically classify activities according to the ontology and an annotation module to annotate the unmatched activities with the existing ontology (ref. Sect. 4.2 *&* 4.3).
- An interoperability model to select the security tools to automate the sequence of activities in an IRP (ref. Sect. 5).

2 Related Work

A large-scale SecOrP requires formalization of the concepts of different security tools and their respective activities. Most of the existing literature on SecOrP only focuses on providing APIs or plugins for multi-vendor tools without considering the importance of formalizing the standard features or concepts used by different tools [1, 5–7]. STIX[1], CyBox[2], and Unified Cybersecurity Ontologies (UCO) are the examples of some of the known ontologies for the security domain. UCO combines the existing ontologies. However, it does not provide an ontology for security tools and their activities; nor does UCO support an IRP's activities, which are required by a SecOrP. A few studies formalize various concepts of information security, threats and attacks related information for sharing the information among security community [15, 16, 18]. Though, none of these studies focuses on formalizing the concepts of IRP or diverse nature of security tools.

One recent study has developed ontologies for enabling tool-as-service (TSPACE) for cloud-platform [17]. Based on stakeholder's requirements and tools artifacts, the

[1] https://stixproject.github.io/about/.

[2] https://cyboxproject.github.io/.

required tools are selected using the ontologies, which help stakeholder to alleviate the semantic conflict while integrating multiple tools. The proposed ontology in TSPACE cannot automate the execution of the activities or enable interoperability among security tools. Moreover, TSPACE does not capture the capabilities of tools essential for *interpretability* and *interoperability*. Our proposed ontological model provides the capabilities of security tools to support *interpretability* and *interoperability* of security tools in a SecOrP. Our work supports the *interoperability* issue by mapping the capabilities of the security tools with the activities of an IRP. Using the ontological model, a SecOrP is able to *interpret* the diverse security tools capabilities for making them work together to *automate* the execution of security tools' activities without any human intervention.

Besides a general lack of *interpretability* and *interoperability* among multi-vendor security tools, we also did not find any work that addresses the issues with changing IRP due to emerging threat behavior. Our proposed prediction module supports the auto-classification of new activity description according to the ontology for automatic execution of IRP. To the best of our knowledge, this is the first work that has enabled *auto-integration* of security tools in a SecOrP and developed a prediction module to classify activity description based on ontology. The *automation* is achieved by enabling *interpretability* and *interoperability* among a variety of security tools from different vendors and *auto-classification* of activity description according to the ontology.

3 Motivation Scenario

An incident is any unwanted event that violates specific security objectives (confidentiality, integrity, and availability) of an organization's assets. An IRP aims to provide the best sequence of activities that are necessary to perform in response to an incident, e.g., alerts for the *phishing email*, *DDoS attack*, and so forth. Table 1 shows an IRP for one such incident, spear phishing email. A phishing email is used to obtain sensitive information by disguising as a trustworthy entity in electronic communication.

Table 1. The incident response plan for a phishing attack

#	Response	Activity Description
ac_1	*Is this* a phishing attack?	*Validate* if this is a phishing attack
ac_2	*Scan* endpoint – malware found?	After running a *scan*, *determine* whether malware was found
ac_3	*Remove* malware – success?	*Determine* whether the malware was successfully removed
ac_4	*Wipe* and *reimage*	If you did not successfully *remove* the malware found, this task instructs you to perform a *wipe* and *reimage* the infected computer
ac_5	*Update* email protection software	If it was determining as phishing attack, you are prompted to *update* email protection software accordingly
ac_6	*Remove* unread phishing email	Perform the steps necessary to *remove* unread phishing email still in the queue

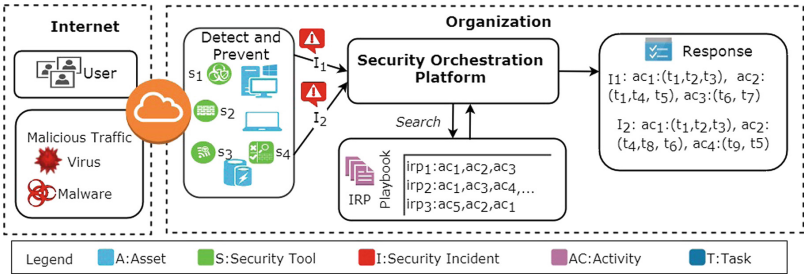


Fig. 1. Overview of a security orchestration platform

Figure 1 shows a scenario of SecOrP where it collects the details of an incident, checks in the playbook for the corresponding IRP and rules therein, select the tools to perform the activities based on the rules, orchestrates the activities and automates the execution of an IRP. Most SecOrPs have a playbook as shown in Fig. 1 where a SOC defines rules based on their respective IRPs. SecOrP shows the scan and ongoing operation through its dashboard based on which a SOC team makes the required decisions, defines new rules in the playbook and performs complex analysis. We refer to the activities that are performed by SecOrP to orchestrate and automate an IRP as *Task*. To address the interoperability issue, an existing SecOrP offers APIs or plugins to communicate with different security tools. Most of these APIs or plugins are not vendors or tools agnostic and fail when updates or changes are required [1, 5, 9]. There are several challenges associated with existing SecOrP; however, in this work, we only focus on the challenges mentioned below. We use the example of Table 1 to illustrate the challenges that arise during the auto-execution of IRP by SecOrP.

First, the IRP of Table 1 is written in text and does not follow a formal structure. There exists ambiguity among different words. Different words are used to define the same types of activities. For example, both *Response* and *Activity Description* of Table 1, i.e., "*Is this a Phishing attack?*" and "*Validate if this is a phishing attack*" are referring to the same activity. A SOC does not follow any specific structure while defining the activities of an IRP. The similar types of activities performed for different security incidents require different tools. For example, "*remove malware*" and "*remove phishing email*" both refer to activity "*remove*" although the execution of these activities requires two different types of security tools. A SecOrP cannot automatically *interpret* the abovementioned similarities or ambiguity.

Second, a SecOrP needs to deal with different tools that are not interoperable to automate the execution of an IRP's activities. For example, to execute an activity ac_1 of Table 1, a threat intelligence platform, e.g., *Malware Information Sharing Platform (MISP*[3]*),* is needed. A *MISP* is used by a SecOrP to validate the incident. The execution of ac_2 requires an *EDR* tool to scan endpoints and *a SIEM* to identify the malware from *EDR* logs. Each activity has one or multiple rules associated with it. A SecOrP uses these rules to orchestrate and automate an IRP by using different security tools. For

[3] https://www.misp-project.org/.

example, if the ac_1 is true, then only it executes ac_2. Based on the results of ac_2, it further executes ac_3 or other activities.

Third, a SecOrP needs to control the flow of the activities performed by different tools. Experts modify the activities based on the tool's availability and preferences. For example, an expert may change one activity description in an IRP from "*analyzing the alert log*" to *"correlating alert log"* after installation of a new IDS in the network router. Installation of a new server requires the security tools' capabilities to fulfil the security requirement of a server. An IRP team defines the plan to protect the server from security incidents. In case, existing tools are unable to provide the required capability; a SOC integrates new security tools to protect the server.

Fourth, there may be multiple tools available for execution of a single activity. For example, different *EDR* tools and dedicated malware detection tools are used to perform "*scan endpoint for malware*." There is a lack of systematic approach that can be followed to perform the selection of interoperable security tools.

Considering the changing activities in IRP that needs integration of new tools, the challenge is how to provide an interoperability model for a variety of security tools to automatically execute different sets of IRP. In the next sections, we first propose the semantic integration framework and then the interoperability model that uses the component of the integration framework to address the abovementioned challenges.

4 Integration Framework for Security Orchestration Platform

4.1 Ontological Model to Enable Semantic Integration

A SecOrP deals with various types of data produced by heterogeneous security tools. These data can be structured, semi-structured, or unstructured. Data produced by one tool are not always interpretable by another tool. Therefore, these heterogeneous security tools are not *interoperable*. We develop an *ontological model* to represent multi-sourced data and enable semantic-based data integration among heterogeneous security tools in a SecOrP [15, 16]. We define the classes of the required ontology by following a structured approach to keep consistency among the classes.

Design and Development of an Ontology Class. We follow a bottom-up approach to develop the main concepts of our ontology that contains three main classes: *SecurityTool*, *Capability*, and *Activity.* These classes are defined to formally represent heterogenous security tools from different vendors. We leverage the TSPACE work [17] to design the capabilities of security tools in-terms of their functional and non-functional features. The functional feature is the ability of a security tool to execute an activity such as *packet capturing*, *intrusion detection* and so forth. The non-functional features include input and output data structures, and configuration details required to execute an activity. For example, a network-based IDS takes *network traffic* or *packet* (i.e., *tcpdump*), where a host-based IDS works with *system logs (i.e., syslog)*. Even though both types of IDSs produce alerts as an output, the *output format* (*i.e., PCAP, CSV*) and data (i.e., *IP address, URL*) also vary depending on SOC's preferences.

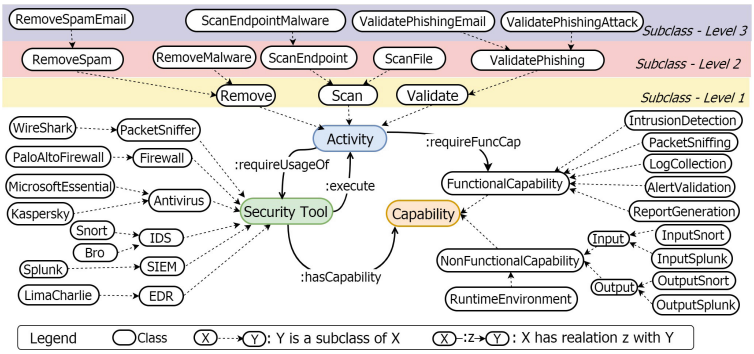


Fig. 2. Excerpt of our ontology (For better quality see at https://github.com/Chadni-Islam/Security-Ontology/blob/master/Ontology.jpeg)

The *Capability* class of the ontology consists of the two subclasses *FunctionalCapability* and *NonFunctionalCapability* to capture the features of security tools as shown in Fig. 2. The diversity among input and output data structures is apprehended using three subclasses under *Non-FunctionalCapability* class: *Input*, *Output*, and *RuntimeEnvironment.* The input and output of the security tools need to be explicitly defined to be analyzed by a SecOrP. A well-designed *Capability* class enables SecOrP to auto-generate the APIs between security tools by retrieving the information about required input commands and produced output. The ability of a SecOrP to deconstruct the output of one tool and then to use the output to formulate the input of another tool enables *interoperability* between isolated security tools.

We analyze the functional capabilities of multiple security tools to identify the subclasses of the *SecurityTool* class, where each tool has more than one functional capability. The *SecurityTool* class is categorized based on the main functionalities of the security tools. We define the first level of the subclass of the security tool based on the types of activities (e.g., *detect*, *monitor*, *scan*, *validate* and so on) they provided. For example, *IDS*, *SIEM*, *Antivirus*, and *Firewall* are different types of security tools that are defined as the subclass of the *SecurityTool* class. The available commercial and open source security tools are categorized under each of these subclasses based on the benchmark of their functional capabilities. For example, different types of SIEM, i.e., *Splunk*[4] and *RSA NetWitness*[5], are subclasses of SIEM.

We define and categorize different types of activities as the subclass of the *Activity* class. The activities are associated with the detection, prevention, recovery and remediation actions of a threat defense and response life-cycle. We follow a systematic set of guidelines to define the subclasses of the Activity class manually. First, we only use the verb and noun of the sentence of activity description to define the subclasses of the Activity class. For example, for the activities of Table 1, *Validate*, *Remove*, *Scan*, *Wipe*, *Reimage* and *Update* are the subclasses of level 1 of the Activity class. Then, we

[4] https://www.splunk.com/.

[5] https://www.rsa.com/en-us/products/threat-detection-response/ .

***ac*₁:** *Is* (**Verb**) *this* (**Det**) *a* (**Det**) *phishing* (**Verb**) *Attack* (**Noun**) *?* (**Punc**) = *Is (Validate) Phishing Attack*
Subclass: *Validate → Validate Phishing → Validate Phishing Email*
***ac*₂:** *Determine* (**Verb**) *whether* (**Adp**) *the* (**Det**) *data* (**Noun**) *associated* (**Verb**) *with* (**Adp**) *this* (**Det**) *is* (**Verb**) *sensitive* (**Adj**) = *Determine Data Sensitivity*
Subclass: *Determine → Determine Data → Determine Data Sensitivity*

Fig. 3. The parts of speech tagging of the incident response plan and removing stop words

combine the adjacent verb, noun, and adjective and discard all other parts-of-speech to define the categories of the subclasses as shown in Fig. 3.

Each subclass of the Activity class has multiple subclasses based on the capabilities required to execute the activity. For example, the execution of two validation activities: *validation of a phishing email* and *validation of exposure of confidential information* require different capabilities; therefore, they are categorized under different subclasses: *ValidatePhishingAttack* and *ValidateDataExposure*. We also consider the activity "*Is this a phishing attack?*" under the class *Validate*, as this is more similar to validating whether an alert/attack is phishing or not. We consider a different sentence with similar meaning into the same class. For example, the activity "*scan endpoint for malware*" and "*scan host for malware*" requires the same types of capabilities and thus are categorized under the same class *ScanEndpointMalware*. These subclasses can have more subclasses depending on the requirements to execute the activities. Figure 2 shows part of the subclasses of the *Activity* class that we have built following the abovementioned process.

Defining Relationships and Constraints. We define the relationship between the classes to select the tools with appropriate capabilities to execute an activity. The relationships between the classes are shown in Fig. 2. We define a set of reasoning rules to enhance the relationships between different classes for error-free integration. These rules enable us to express conditions about the occurrence or non-occurrence of the required activities, the creation of instances, tracking and managing activities of a SecOrP. For example, each security tool must have at least one functional capability associated with threat defense and incident response to execute an activity. The security tools must satisfy the capabilities associated with a class to be part of that class.

Execution of each activity depends on the availability of the relevant security tools and preference of an organization's security requirements. An auto-execution of an activity requires at least one tool with the required functional capability to execute a desired activity. We impose different types of restrictions for creating the instance of a class that must satisfy the relationship it holds with other classes. The defined rules enable a SecOrP to avoid ambiguity while creating an instance of a class. A SecOrP executes the activities sequentially; as a result, the security tool that is selected to execute ac_{i+1} must have access to the output of a security tool that executes ac_i. For example, if *Splunk* requires to analyze the alert log produce by *Snort*[6], it must have

[6] https://www.snort.org/.

access to the output file of *Snort*. Similarly, a SecOrP needs to have the authorization to run and stop every security tool that is integrated into it.

The proposed ontological model enables a SecOrP to interpret activities and security tools capabilities. Retrieving the information of the non-functional capability class, SecOrP can *interpret* the data generated in various forms and also formulate the input command to invoke a particular tool for auto-execution of the activity.

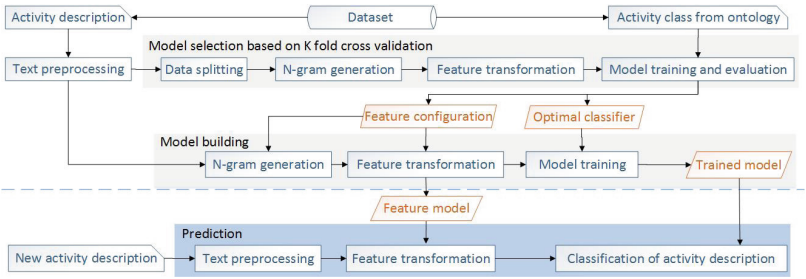


Fig. 4. Development of the prediction module

4.2 Classification of Activities Based on Text Similarity

A SOC adds new types of activities or updates the existing IRP to keep playbook updated for emerging threat. Considering the available tools to execute IRPs, we leverage existing NLP and ML techniques to automatically classify the new activity description according to the activity ontology. It makes SecOrP capable of analyzing an IRP and transforming the data into a representation that gives both an analyst and machine insights about the data. We consider the classes of Activity class in different level separately (Fig. 2). An example of a class on each level includes: level 1 {*Remove*, *Scan*, *Validate*}, level 2 {*RemoveSpam*, *RemoveMalware*, *ScanFile*}, and level 3 {*RemovePhishingEmail*, *ValidatePhishingEmail*}. From the perspective of ML, this problem is designed as a multiclass supervised text classification problem.

Given a new activity description in an IRP, we design the prediction module to classify the activity description according to the classes of the ontology. The overall workflow of building an ML-based *prediction module* is given in Fig. 4. The dataset is consist of the activity descriptions labeled according to the ontology. Table 2 shows examples of the labels that correspond to the activity described in each level of the ontology for *Activity* class. Initially, the dataset is divided into training and testing set. The key components of building the ML model includes *text preprocessing*, *model selection*, *model building*, and *prediction*. The model selection and model building processes work on the training set and the prediction process work on the testing set or with new activity description.

Text Preprocessing. We start with a corpus of activity description and follow the standard process of text wrangling and pre-processing. During the preprocessing step, we remove the null-value, punctuation, stop words, and meaningless words for the analysis. We perform parts-of-speech tagging of the text before removing the stops words and only keep the verb, adjective, and noun.

Table 2. Activity description and corresponding class label

Activity description	Level 1	Level 2	Level 3
Scan endpoint to see whether malware was found	*Scan*	*ScanEndpoint*	*ScanEndpointMalware*
Is this a phishing email	*Validate*	*ValidatePhishing*	*ValidatePhishingEmail*
Isolate the malicious node from the network	*Isolate*	*IsolateMalicious*	*IsolateMaliciousNode*

Model Selection. We use the preprocessed text to perform *k-fold cross-validation* to select the optimal classifiers for the prediction module. As shown in Fig. 4, the model selection method has four steps: *data splitting*, *n-gram generation*, *feature transformation*, and *model training and evaluation*. The preprocessed text in each fold is split in the training set and validation set of equal sample size. We generate *word-based n-gram* for the training and validation set that are merely the combinations of adjacent words of length n. We combine the n-gram with the Term Frequency-Inverse Document Frequency (TF-IDF) for each activity description.

The ML-based classifiers cannot directly process the text documents. Most of them expect numerical feature vector of fixed size whereas the raw text documents are of variable length. The features generated from n-gram are presented into *Document-Term Matrix (DTM)* where each row corresponds to an activity description and each column correspond to a word in the term.

In the model training and evaluation steps, we train the four classifiers (*Random Forest*, *Linear Support Vector*, *Multinomial Model* of *Naïve Bayes*, and *Logistic Regression*) on the training set and then evaluates the model on the validation set using different evaluation metrics (*accuracy*, *recall*, *precision*, and *f1-score*). The classifier with the highest average cross-validation score is selected as an optimal classifier. The process is repeated for each level (level 1, 2 and 3). The optimal classifiers and feature representations are returned for all three levels.

Model Building. The model building process uses the whole set of preprocessed training set to generate the word n-gram. Here n-gram generation and feature transformation are based on the identified feature configuration for each level of class. The generated n-gram vocabularies are combined with the feature configuration to create the feature model. The feature model has been saved to transform the data for future prediction. The extracted features are trained with the optimal classifiers returned in the model selection process to build the prediction model for each level.

Prediction. The prediction process is used for both testing the trained model and classifying the new activity description. In this process, the activity descriptions are first preprocessed and then using the saved feature model transformed to a feature set. Finally, the features set is used by the saved trained model to determine the class of the activity description in terms of the ontology for each level.

The prediction module reduces the manual analysis of the activity description by classifying the activities according to the ontology.

4.3 Design and Development of the Annotation Module

The new activity description may not always fall in any of the existing activity class of the ontology. In this context, we are considering these types of description as an outlier. To identify the outlier description, we perform text-based similarity checking of the updated or new description with the existing activity description and measure the cosine similarity. We define a threshold for considering whether the description is an outlier in terms of the existing set of activity description. If the new description is not an outlier, then only the description is sent to the prediction module. If the new description is considered as an outlier, we develop the annotation module to automate the generation of the possible list of classes following the same set of guideline that is proposed to design the *Activity* class in Sect. 4.1. The generated classes are matched with the existing set of classes, and if none of the classes are found in the ontology, the annotation module recommends the possible list of the classes to a user. Once a user selects the corresponding classes, it creates new classes for the activity description and if required requests for additional details about the classes from the user to keep the ontology consistent.

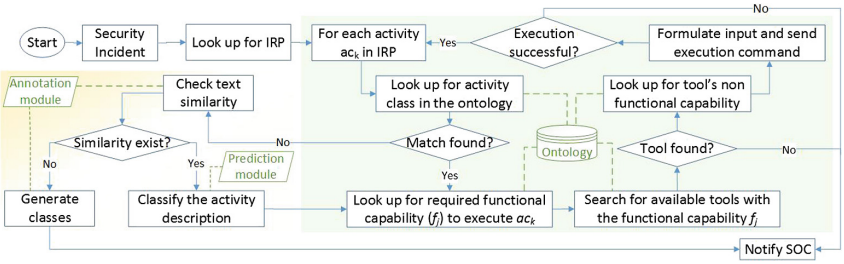


Fig. 5. Workflow of the proposed solution

5 Interoperability Model for Execution of IRP

A SecOrP may need to invoke a different set of security tools in a different order to execute a variable sequence of IRPs. For example, one IRP may include an activity *scan endpoint*, followed by another activity *correlate alerts log*, whereas another IRP may include *correlating alerts logs* followed by *scan endpoint.* Both of these IRPs require the same security tools in different orders. We provide the interoperability model for auto-execution of the required IRPs, where one tool can understand the output of other tools. The model also helps SecOrP to interpret the output and input of different security tools. For example, a SIEM tool needs an output of alerts produced by IDS and a system log produced by EDR to perform correlation. Figure 5 shows the overall workflow of the interoperability model starting from gathering a security incident to notify a SOC. Two key tasks of the interoperability model are: select the desired tools based on their functional and non-functional capabilities and invoke the tools to execute an IRP. The key components of the integration framework (*ontological*

model, *prediction* and *annotation module*) as shown in Fig. 5 are used to design the interoperability model.

We have designed a Query Engine (QE) to retrieve the information from the ontology. Given a set of Security tools $S = \{s_1, s_2, ..., s_m, ...\}$, a list of the required activities $AC = \{ac_1, ac_2, ..., ac_k, ...\}$ and a list of capability, $F = \{f_1, f_2, ..., f_j, ...\}$, a SecOrP looks up for the corresponding IRP for each security incident. For each activity ac_k of *IRP*, SecOrP invokes the QE to search for the corresponding *Activity* class. If the activity is found in the ontology, the SecOrP invokes QE to retrieve the capability required to execute the activity. Considering f_j is the required functional capability, a SecOrP queries to retrieve the security tool that has the functional capability, f_j. In case, multiple tools are available, the SecOrp selects the right tool from the list. In the next step, the SecOrP retrieves the non-functional capability of the selected security tool to formulate the input command for instructing the tool to execute the activity. The QE extracts the necessary information from the ontology to formulate the input for the tool. After constructing the input command, a SecOrP calls the tool's corresponding routine to execute the activities. If the execution is successful, the next activity in the IRP is executed by following the same sequence of tasks (performed by SecOrP).

Considering the output produced by one tool s_m is provided as an input to another tool s_p, a SecOrP checks for the interoperability of the two security tools. The SecOrP deconstructs the output of s_m to formulates the input of the s_p. It is only possible if the tools are interoperable; otherwise, the SecOrP notifies the SOC.

An activity's description may change continuously; if no class is found for a particular activity, a SecOrP first invokes the AU unit to determine the possibility of the new description to be part of the existing ontology. Based on the similarity measurement it either generates the list of classes or invokes the prediction module to classify the activity description. After getting an appropriate class from the prediction module, the same steps of looking for the required functional capability and non-functional capability to execute the activity are carried out.

Following the abovementioned process, a SecOrP can automate the sequence of activities in an IRP even when changes occur in the underlying execution environment. The interoperability model enhances the capability of a SecOrP to automate the execution of an IRP by interpreting the activity, required capability, and tools interoperability.

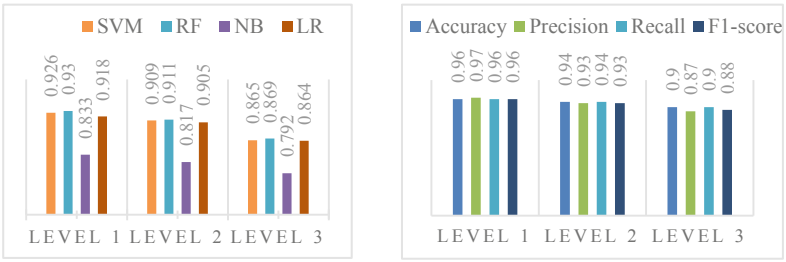
6 Experiments and Results

We carried out a set of experiments to assess the feasibility of the proposed prediction module and interoperability model.

Preparing the Dataset for Prediction Module: Our experimental dataset is based on the IRP crawled from the website of *ServiceNow*[7] that resulted in 1080 activity

[7] https://docs.servicenow.com/: offers on-demand, cloud-based IT service management solution, forms-based workflow application development, automation workflow, productivity tool for business user and so forth.

descriptions. For each activity description, we manually labeled the classes according to the ontology as shown in Table 2. We have *34 categories* under level 1, *67 categories* under level 2 and *74 categories* under level 3.



(a) Validated Weighted average of F1-Score **(b)** Testing results of Random Forest classifier

Fig. 6. Bar plot of (a) validated weighted average of F1-score for optimal configuration of different classifiers and (b) testing results of random forest for three levels of class

Implementing the Prediction Module: We used the *sci-kit-learn*, *NLTK* and *spaCy* package of python to build a classifier. For each level, we first separately implemented four classification algorithms with different hyper-parameter settings. We performed k-fold cross validation for each configuration by splitting the data set into different training and validation sets. We used the function *GridSearchCV()* to select the optimal configuration and perform cross-validation for each classifier. For both *Support Vector Machine (SVM)* and *Linear Regression (LR)*, we considered different values for the regularization parameter (i.e., 0.01., 0.1, 1, 10, and 100). For *Multinomial Naïve Bayes (NB)*, we considered prior probability of class True and False. For *Random Forest (RF),* we considered different values for estimators (i.e., 10, 100, 20, 200, 50, and 500) and the maximum number of leaves (i.e., 10, 50, 100, and 200). Figure 6(a) shows the results of different classifiers for the optimal configuration. We examined the performance of the classifiers in terms of accuracy and F1-score [19]. F1 score is considered more reliable than accuracy. *Accuracy* reflects the total of the correct predictions divided by the total number of cases. F-1 score is the harmonic mean of the Precision and Recall. *The precision* represents the total of the correct predictions for each class divided by the total number of activities predicted for that class. The *recall* is the correct prediction for each category divided by the total number that belongs to this category. Comparing the results of the classifier, we found that RF outperformed other classifiers. We built the final model with the RF classifier. The optimal configuration for *RF* (estimators, maximum leaf) for levels 1, 2 and 3 are (50, 100), (100, 100) and (10, 200), respectively. We used 70% of the activities for each level as the training data and 30% as the testing data. Figure 6(b) shows the results of the RF for different evaluation metrics.

Developing the Interoperability Model: We implemented a Proof of Concept (POC) system using seven security tools (*Snort, Splunk, LimaCharlie, Wireshark, WinPcap, Microsoft essential,* and *MISP*) to study the viability of the interoperability model. We described their capabilities in terms of the ontological model and used a list of IRPs with different activities. We used the network traffic and system logs as the

input to identify the security incidents. The experimental study used 21 different capabilities and 9 IRPs with 17 activities. We only considered the activities for which the capabilities were available. We changed the activities and observed the corresponding changes in the operation's execution.

Discussion: The results showed that in more than 90% cases (Fig. 6(b)) the prediction module accurately classified the activity descriptions. The performance in classifying the activities in level 2 and 3 is lower than that in level 1. The reason for this appears to be the number of members in these classes is lower than that in level 1. The more input data we can provide to the classifier the more accurate results it will produce. Besides, the activity description was passed to the prediction module only when the text similarity was found, which makes the classifiers less error-prone towards the new activity description that does not belong to any of the existing classes.

Out of the 17 IRPs, the POC was able to automate 15 IRPs successfully and 2 IRPs partially. While modifying the activity descriptions, there were two activities (update email protection software and detect phishing email) for which security tools were not available. For these two activities, the interoperability model was unable to find suitable security tools, thus failed to automate the execution of that particular IRPs. Except for these two activities, the POC automatically (a) retrieved the information from the developed ontology; (b) generated the configuration details to call the desired security tools; and (c) thus enabled *interpretability* and *interoperability* among different security tools and SecOrP.

Threats to Validity: We developed the ontology based on freely available and open source security tools' capabilities, and activity descriptions, which might not fully represent the situations of scenarios of an organization. Considering the development of an ontology is an incremental process, a human expert can easily extend the ontology to incorporate the tools used in an organization. The selected optimal may not guarantee the highest performance for classifying the new and updated activity descriptions since an infinite number of configurations are available to tune the hyper-parameters of ML classifiers. The selected classifiers might not be the best one, but it provides a learning-based approach to classify the activity description which can be further improved and extended with different classifiers and configurations. The model we built is retrainable and can be easily trained with the new dataset.

7 Conclusion

Given the widespread adoption of SecOrP over the last couple of years, there is an increasing demand for self-adaptive SecOrPs. Our research purports to devise a solution that can enhance the *interpretability* and *interoperability* of security tools integrated into a SecOrP. The proposed approach allows a SecOrP to select the required security tools that are interoperable for auto-execution of an IRP. We have introduced an ontological model to formalize the security tools, their capabilities, and the activities of an IRP. A learning-based prediction module is proposed to reduce the manual work of security staff to define the classes for activity in a playbook. The proposed interoperability model successfully automates the execution of most of the IRPs at runtime.

In future work, we will extend the system to automate the generation of the APIs from the ontology. We also aim to use the semantic definition of tools capabilities to auto-create the APIs when new security tools with new capability are integrated, and design a probabilistic model for selecting and integrating security tool.

Acknowledgment. This work is partially supported by Data61/CSIRO, Australia. We acknowledge the contributions of the shepherd reviewer Professor Andreas L. Opdahl from the University of Bergen, Norway who provided insightful comments with continuous engagement to improve the paper.

References

1. Demisto. https://www.demisto.com/wp-content/uploads/2017/04/MH-Demisto-Security-Automation-WP.pdf. Accessed 11 Oct 2017
2. Koyama, T., Hu, B., Nagafuchi, Y., Shioji, E., Takahashi, K.: Security orchestration with a global threat intelligence platform. NTT Tech. Rev. **13**, 1–6 (2015)
3. Luo, S., Salem, M.B.: Orchestration of software-defined security services. In: 2016 IEEE International Conference on Communications Workshops, ICC 2016, pp. 436–441 (2016)
4. Enterprise Strategy Group. https://www.esg-global.com/research/esg-research-report-cybersecurity-analytics-and-operations-in-transition. Accessed 25 Feb 2019
5. McAfee. https://www.mcafee.com/au/solutions/orchestration.aspx. Accessed 20 Oct 2017
6. Komand. https://www.komand.com/. Accessed 21 Oct 2017
7. Feitosa, E., Souto, E., Sadok, D.H.: An orchestration approach for unwanted internet traffic identification. Comput. Netw. **56**, 2805–2831 (2012)
8. SIEMPLIFY. https://www.siemplify.co/security-orchestration-automation. Accessed 1 Nov 2017
9. SWIMLANE. https://swimlane.com/use-cases/security-orchestration-for-automated-defense/. Accessed 20 Nov 2017
10. Yu, T., Fayaz, S.K., Collins, M., Sekar, V., Seshan, S.: PSI: precise security instrumentation for enterprise networks. In: Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA (2017)
11. SWIMLANE. https://swimlane.com/ebook-sao-capabilities/. Accessed 20 Oct 2017
12. FireEye. https://www.fireeye.com/solutions/security-orchestrator.html. Accessed 11 Jan 2018
13. Microsoft. https://www.microsoft.com/en-us/windowsforbusiness/windows-atp. Accessed 21 Jan 2018
14. Crowley, C., Pescatore, J.: The Definition of SOC-cess? SANS 2018 Security Operations Center Survey. SANS (2018)
15. Evesti, A., Ovaska, E.: Ontology-based security adaptation at run-time. In: 2010 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 204–212. IEEE (2010)
16. Syed, Z., Padia, A., Finin, T., Mathews, M.L., Joshi, A.: UCO: a unified cybersecurity ontology. In: AAAI Workshop: Artificial Intelligence for Cyber Security (2016)
17. Chauhan, M.A., Babar, M.A., Sheng, Q.Z.: A Reference architecture for provisioning of tools as a service: meta-model, ontologies and design elements. Future Gener. Comput. Syst. **69**, 41–65 (2017)

18. Krauß, D., Thomalla, C.: Ontology-based detection of cyber-attacks to SCADA-systems in critical infrastructures. In: 2016 6th International Conference on Digital Information and Communication Technology and Its Applications, DICTAP 2016, pp. 70–73 (2016)
19. Dua, S., Du, X.: Data Mining and Machine Learning in Cybersecurity. Auerbach Publications, Boca Raton (2016)