Journal of Universal Computer Science, vol. 25, no. 9 (2019), 1066-1088
submitted: 31/1/19, accepted: 9/7/19, appeared: 28/9/19 © J.UCS

A Context-based Defense Model for Assessing Cyber Systems' Ability To Defend Against Known And Unknown Attack Scenarios

Yosra Lakhdhar
(Communication Networks and Security Research Lab.
SUPCOM, University of Carthage, Tunis, Tunisia
lakhdhar.yosra@gmail.com)

Slim Rekhis
(Communication Networks and Security Research Lab.
SUPCOM, University of Carthage, Tunis, Tunisia
slim.rekhis@gmail.com)

Noureddine Boudriga
(Communication Networks and Security Research Lab.
SUPCOM, University of Carthage, Tunis, Tunisia
Computer Science Department, University of Western Cape
Bellville, South Africa
noure.boudriga2@gmail.com)

Abstract: Presently, attackers succeed to damage different cyber systems no matter whether cyber security solutions are implemented or not. This fact can be explained by the information insufficiency regarding the attack environment and the deployed solutions, in addition to the predominant use of pre-built cyber attack databases, making the supervised system incapable of defending itself against zero-day attacks. We present in this paper an enhanced cyber defense model to assess the effectiveness of the deployed security solutions to defend against potential generated attack scenarios under various contexts (the configuration of distributed security solutions, named observer agents, the type and location of reaction systems, and the type of data visible by the deployed solutions). Furthermore, we propose a model ensuring the generation of known and unknown attack scenarios starting from the formal description of system variables and their interactions. In addition, we develop the concept of observable executable scenario that ensures the step by step observation of attack scenarios execution, the assessment of observer agents' reactions, and the detection of attack occurrence in a distributed system. The results of the conducted simulations using real case studies are presented to exemplify the proposal.

Keywords: Cyber defense, security assessment, distributed agents, unknown attacks, formal generation, model checking
Categories: C.2.0, K.6.5, L.4.0

1 Introduction

Faced to the enhanced released cyber attacks, securing cyber systems becomes a serious worldwide problem. Regardless of the adopted defense strategies, many attacks are actually occurring across the world such as, the NotPetya ransomware attack that took

place on June 2017 which was described by the White House as "the most destructive and costly cyberattack in history"[1] These attacks and their catastrophic impact on different critical systems proved the incapability of traditional security solutions to protect our systems. Therefore, it is mandatory to develop new tools to efficiently react to malicious actions. These techniques have to accurately model the cyber system, ensure a proactive cyber defense and predict the infrastructure reaction faced to various attacks.

The biggest challenge that faces cyber defenders is that, no matter whether the attack scenario is known or not, cyber attacks continue to take place even if security solutions are deployed [Razzaq et al., 2013] [Hina and Dominic, 2016]. This fact can be explained by three major points. First, the used defense strategies are usually based on a pre-built attack scenarios database, which makes the cyber system incapable of dealing with unknown attack scenarios. Second, even if the scenario is known, the security solutions may not detect it owing to the lack of data about what is actually occurring in the system as they are unable to completely observe, collect and analyze all the system assets. Finally, usually a security solution reaction is localized, whereas the attack may impact various system components located everywhere, making the security reaction useless.

The research works that dealt with the assessment of cyber system to defend against attack execution under various contexts, can be classified into three classes: a) the first class is related to the assessment of cyber systems' security [Plosz et al., 2017] [Li et al., 2017] [Berenjian et al., 2016] [Si-chao and Yuan, 2016] even with the presence of zero-day attacks [Razzaq et al., 2009] [Kotenko and Chechulin, 2012].

[Keramati, 2016] [Joshi et al., 2018]. However, none of these works studied the cyber infrastructure reaction to attacks at step by step of their execution; b) the second group is related to cyber defense system architecture. The use of a centralized approach [Lakhdhar et al., 2016] [Guezguez et al., 2017] [Vasilomanolakis et al., 2015a] presents two major problems which are scalability and existence of a Single Point Of Failure. Thus, the proposed model is a distributed one; and c) the third group is related to the introduction of context in cyber defense models [Lakhdhar et al., 2018b] [Aparicio-Navarro et al., 2017b] [AlEroud and Karabatis, 2017] [Giura and Wang, 2012]. Although none of these works defined the safety conditions of the deployed solutions in the considered context.

The supervised system will be modeled as a set of variables representing the parameters to be monitored, a library of atomic actions defining the variation of variable values from one state to another, and a system constraints library presenting the relations to be satisfied between these variables. Based on this system model, all the possible scenarios will be generated, and the concept of local executable scenario will be developed to describe the scenario execution at one part of a distributed system. After modeling the cyber system and generating all the possible scenarios, we will define, model and deploy a set of security solutions (observer agents) to supervise system parameters, where each agent is characterized by an observation function generating different observation data based on a predefined security condition predicate ensuring the proper functioning of these agents, and a reaction database defining the countermeasures that it can execute once a malicious event is detected. As based on the

[1] https://www.whitehouse.gov/briefings-statements/statement-press-secretary-25/

environment in which it is executed the intruder may or may not damage the supervised system, we will define the attack execution context as the set of deployed security solutions and their configurations and we will introduce the concept of observed local scenario to describe both the observed part of the generated scenario and the agent' behavior and reaction to each execution step. Consequently, our model consists in: (a) modelling the supervised cyber system as a set of variables presenting the system parameters to be monitored. To generate all potential scenarios, we define a set of libraries. In fact, we introduce a library of atomic actions presenting how the system variables values change from one state to another. In addition, a library of system constraints describing the relations to be satisfied between these variables is introduced to remove the unacceptable generated states; (b) developing an algorithm allowing to generate all the possible known and unknown attack scenarios by generating successive states using the defined libraries. These scenarios are outlined by the violation of the safety properties which are modeled as predicates computed over the generated observations by the observer agents network on each execution step; (c) modeling the security solution as a set of distributed observer agents deployed to monitor the different system parameters where each agent is characterized by its observation function, defense domain and reaction database; and (d) contextualizing attack execution by defining the concept of context to describe the security environment in which the attack takes place. Using this context, we develop the concept of observed executable scenario that allows to observe a step-by-step execution of an attack scenario together with the observer agents' reactions.

Our contribution is three-fold. First, we develop a theoretical model for the generation of both known and unknown scenarios starting from libraries of atomic actions and system constraints. Using such a model, an action is modeled as the simultaneous execution of a set of atomic actions, ensuring the modification of the system status (i.e., movement from one state to another). To ensure the coherence of the generated states, we added a set of system constraints to be verified in each step of the scenarios generation process. Second, we develop a model for generating executable scenarios in a distributed and networked environment. In fact, by defining the local executable scenario as the execution of a global scenario at one part of a distributed system, and by considering the observer agent behavior and reaction to each executed step, the proposed model can do a step-by-step assessment of system ability to defend against executed attacks and identify the step at which the scenario can be blocked. Third, we introduce the concept of attack execution context and prove its importance in assessing the system resilience to different attacks execution. The context is modeled as the set of deployed security solutions and their configurations, in addition to the security conditions ensuring their proper functioning.

This work is an extension of the context-based model presented in [#Lakhdharr2018]. The extension can be described as follows: First, we do not rely on the existence of a database of known attacks which make the system able to deal with zero-day attack scenarios. The model we are proposing allows us to progressively generate these actions by referring to system variables and the relations between them. Second, we extend the concept of cyber attack execution context by better describing the security solutions, especially by adding the conditions under which they can observe the system behavior.

The remaining part of this paper is organized as follows. Section 2 will be dedicated to the state of the art. In Section 3, we present the requirements of a context-based defense model for assessing the system ability to defend against known and unknown attack scenarios. The cyber system modeling is presented in Section 4. We detail in Section 5 the approach used to generate potential scenarios and we introduce the concept of global and local executable scenarios. In Section 6, we model the deployed security solutions and detail their different parameters. The attack contextualization will be presented in Section 7. Our methodology is detailed in Section 8. Before concluding the paper, the results of implementing the approach using real case studies are presented in Section 9.

2 State of the art

We present in this section the state of the art related to security solutions assessment, cyber defense model architecture, and attack contextualization.

2.1 Works related to security solutions assessment

Many works were developed to analyze and assess the system security state:

In [Plosz et al., 2017], the researchers proposed a method combining security and safety assessment techniques for industrial collaborative automation systems. For security assessment, researchers started by modeling the system using the Data Flow Diagram method (DFD). Then, they used ETSI TVRA (Threat, Vulnerabilities, and implementation Risks Analysis) method to assess the risk by computing its likelihood. Although, the proposed model does not show the real-time reaction of the system to each executed action.

In [Li et al., 2017], the researchers proposed a State-Aware Risk Assessment Model (SRAM) that considers the system state information in addition to IDSs' output to enhance the accuracy of Intrusion Response Systems. However, researchers did not present how the IDS output will be fused with the State Index value, and whether their model can handle different types of detection systems.

In [Berenjian et al., 2016], the researchers introduced an Automated Intrusion Response System (AIRS) which consists mainly of a Web IDS, a fuzzy system and a response module. This work focused only on web-applications security and did not include the characteristics of the deployed WIDS.

In [Si-chao and Yuan, 2016], the researchers presented a risk assessment method based on both attack graph model and Hidden Markov Model (HMM). Although, this method did not show the attack context and the system reaction to the attack execution.

Security assessment considering unknown attacks was also studied by different researchers. For example, in [Razzaq et al., 2009], researchers developed a multi-layered defense approach for the detection of both known and unknown web application attacks. Although, the proposed approach is not a generic one as it deals only with web applications attacks.

In [Kotenko and Chechulin, 2012], researchers proposed an Attack Modeling and Security Evaluation Component (AMSEC) to be integrated in SIEM (Security Information and Event Management) systems. The presented component contains a “security level evaluator” component to generate attack graphs, analyze known and

zero-day vulnerabilities, and evaluate the supervised system security level. Nevertheless, the researchers neither presented the used evaluation metrics, nor showed how they can be computed.

In [Keramati, 2016], researchers presented a graph-based approach for risk assessment of zero-day attacks. In fact, they defined a set of parameters to be able to deduce the risk of known and unknown vulnerabilities. It is true that this work studies the risk of zero-day vulnerabilities but it did not assess the system ability to defend against them.

In [Joshi et al., 2018], researchers proposed a three-layered framework to assess the risk of zero-day vulnerabilities. The first layer is a zero-day path generator where an attack graph will be generated by detecting abnormal activities. The second layer is a risk analyzer one, where the severity of the generated nodes will be ranked using AttackRank algorithm. The last layer is a physical layer used to store collected and generated information. Researchers did not present the countermeasures and their impact on the generated graph to show the system ability to defend against generated attack paths.

It is true that cyber system security assessment was studied by many research works, but a general model for assessing system ability to defend against various attack scenarios where a formal definition of security solution is presented and the supervised system reaction to cyber attacks at step by step of their execution is considered, still needs to be developed.

2.2 Works related to cyber defense systems architectures

Cyber defense systems can be divided based on their architectures to distributed and centralized ones. In this subsection, we will cite a set of works presenting the two architectures and their characteristics.

In [Lakhdhar et al., 2016], researchers designed a centralized graph-based ACD (Active Cyber Defense) model and developed a set of analytics to describe both the protected cyber systems and the deployed security solutions. In this work, the security reaction is executed based on the attack value that should not exceed a predefined threshold, where the decision is taken by a centralized Cyber Defense Center.

In [Guezguez et al., 2017], the researchers presented a centralized observation-based technique ensuring the detection of various cyber attacks on femtocells. In fact, they installed observer agents on the clients' smartphones, whose main role consists in monitoring, collecting, and sending detected radio sensitive events to a centralized Observer Management Server that will analyze the received data to detect attacks from femtocells.

In [Vasilomanolakis et al., 2015a], the researchers developed an open-source centralized honeypot-based cyber incident monitor called TraCINg (TU Darmstadt Cyber Incident moNitor). The proposed system receives various alerts data generated by distributed deployed honeypot sensors, aggregates and correlates these alerts and provides a central visualization interface.

Using centralized architecture presents a scalability problem, as it is not obvious that one central server can analyze a huge amount of data in real-time. Also, these systems suffer from SPOF (A main part of a system that once it fails, the entire system will stop working) as once the central agent is compromised, the system becomes inefficient. For these reasons, other cyber defense architectures were also proposed in

the literature. For example, in [Vasilomanolakis et al., 2015b] researchers proposed an open-source distributed collaborative IDS named "SkipMon". The proposed system uses the P2P overlay SkipNet for the communications between the deployed IDSs and gossiping technique to circulate alerts between them. The problem with CIDS is that sharing and exchanging data between different sensors can lead to high resources consumption besides a communication overhead. For that, many researchers used game theory to deduce the optimal configuration to be adopted for the deployed IDS. For example, researchers in [Ghorbani et al., 2016] modeled the interaction between attackers and the deployed IDS in a CIDS framework with a nonzero-sum stochastic game. Moreover, they used the solution concept of Stationary Nash Equilibrium to describe the optimal stationary strategies of defenders and to predict the attacker's behavior.

Different cyber defense system architectures (centralized, distributed and collaborative) were presented by researchers, but as the executed attack scenarios may impact various system components located everywhere, a model ensuring the generation of executable scenarios in a distributed and networked environment, still needs to be developed.

2.3 Works related to attack contextualization

Protecting cyber systems from the released attacks requires the contextualization of their executions. The concept of "context" was studied by many researchers in various fields [Wan, 2009] [Snidaroa et al., 2015] [Aleroud and Karabatis, 2017]. For cyber defense systems, cyber attack contextualization was studied differently by researchers. For example, in [Lakhdhar et al., 2018b], the context was defined as six tuple information used to calculate different probabilistic measures to proactively assess attack damages. In fact, the context was used to compute the probability of each scenario execution, in conjunction with other metrics used to trigger a security reaction.

In [Aparicio-Navarro et al., 2017b], the researchers proved the extent to which the introduction of contextual information in an Intrusion Detection System enhances its performance. In fact, they used the network Pattern-of-Life (PoL) as a source of contextual information. To integrate the latter in the intrusion detection process using Fuzzy Cognitive Map (FCM), they presented two different approaches. The first consists in using the FCM output to construct a new metric to be fused later by Dempster-Shafer (D-S) theory, whereas the second approach consists in using the contextual information to adjust the output of the D-S fusion process. The integration of contextual data prior to the data fusion process was also studied in [Aparicio-Navarro et al., 2017a]. In the FCM construction, researchers studied the use of three metrics which are Throughput, Communication Rate and Destination Port Distribution.

In [AlEroud and Karabatis, 2017], researchers designed a context-based framework to be used in conjunction with IDS to improve the detection accuracy of cyber attacks. They generated Semantic Link Networks presenting the contextual relationships between different attacks, and adjusted it by integrating domain knowledge extracted from taxonomies about cyber attacks. Moreover, they defined attack contextual profiles using activity features of connections to filter the non-relevant generated predictions about malicious activities.

Moreover, researchers in [Giura and Wang, 2012] proposed a context-based framework for the detection of Advanced Persistent Threats (APT) attacks. An attack

pyramid model was introduced where the goal of the attack is at the top of a pyramid, and the different environments where the events related to this attack occurred define the lateral planes. These events are correlated later into contexts and analyzed by an alert system responsible of triggering security reactions. Formally, a context was defined as seven-tuple information including the correlated events, the attack confidence indicator and the risk level.

Attack execution context was defined differently by researchers. However, a formal model of the latter describing the environment in which the attack is executed by considering among others the deployed security solutions and their configuration and the security conditions of their proper functioning, has not been yet developed.

3 Functional requirements

A context-based defense model for assessing systems' ability to defend against known and unknown attack scenarios should fulfill the subsequent requirements:

Develop a virtual execution technique: It is important to develop a technique allowing to verify the system resilience to the execution of different known and unknown attack scenarios. This technique needs to ensure the generation of all possible scenarios without referring to a static cyber attacks library.

Contextualize the attack execution: The impact of an attack scenario execution varies from one system to another with respect to different parameters as the network topology and the deployed security solutions which need to be considered to accurately assess the security solution's response to their execution.

Model the security solution observations: The information observed and collected by the cyber security solutions are characterized by their incompleteness, which make it crucial to develop a model ensuring the execution of security reactions based on these observations.

Model the reaction system: Developing an accurate defense model requires, besides modeling the supervised system and the security solutions, to model the security reactions. Indeed, a formal description of the latter and the conditions under which they can be triggered have to be properly specified.

Model the observed attack scenario execution: The attack scenario can be defined as a set of actions moving the system from one state to another until an unsafe state is reached. This definition implicitly incorporates the system infrastructure reaction to each execution step. Modeling the real attack execution needs to consider the executed actions, the distributed security solutions, the incomplete generated observations and the possible triggered reactions.

4 Cyber system modeling

In this section, the system model and the used libraries will be presented.

4.1 Library of atomic actions AtAc:

In our model, a cyber system will be modeled by a set of variables $Var = \{v_1, v_2, \dots, v_n\}$ where each variable v_i defines a parameter to be monitored. To identify the set of required variables, one of the following three main approaches could be followed. The first considers the use of an available formal specification of the system to identify the system variables and deduce the types of modifications that they can incur. The second considers the use of the system as a blackbox to study its output/behavior with respect to different inputs, and consequently infer a list of system variables and the potential types of modifications that they can have. The third considers the use of an informal specification of the system to study its properties and functionalities and thus to deduce the set of variables that can be used to model it. To prove the completeness of the set of modelled system variables, one could prove that for every identified system behavior, at least one of the system variable must change its value.

The valuation of these variables defines a system state s that we model as $s =< ||v_1||_s, \dots, ||v_n||_s >$ where $||v_i||_s$ presents the value taken by variable v_i in the system state s. From one state to another, the value of one/many variables changes with action execution. In this work, we assign to each variable v_i a function f_{v_i} defining a relation between values of variable v_i in two successive system states s_i and s_{i+1}. As an example of function f_{v_i}, we can cite: f_{v_1} (s)=$||v_1||_s + 1$ which means that from one state to another the value of v_1 should increase by 1. The succession of system states defines a scenario execution w which we model as w=$< s_1, s_2, \dots, s_m >$ where m presents the number of states composing the executed scenario.

4.2 System constraints library SysCons

We define SysCons as a library presenting the relations to be satisfied between the variables in Var that our model will use to remove the improper state (i.e., a state where the defined relations are not satisfied). From the set of defined relations, we cite the following ones:

The variable domain function "Dom{v}": Dom{v}=$\cup_i\{Pred_i\}$ / $\forall s: Pred_i(s) = True$; where $Pred_i$ is a predicate defined using one variable v and computed on a state s. These predicates define the admissible values by v in any generated state obtained using the library AtAc. In fact, any variable $v_i \in Var$ has limits that it must not exceed along the scenario execution, otherwise, the model can deduce that the reached state is an unacceptable one.

The state collision between variables "SColv$\{v_i, v_j\}$": SColv$\{v_i, v_j\}$= $\cup_k\{Pred_k\}$ / $\forall s: Pred_k(s) = False$; where $Pred_k$ is a predicate defined using two or many variables and computed on a state s. These predicates define the collision set

between these variables' values in the same state s (i.e., the values that must not be taken by a set of variables in the same state s).

The execution collision between variables "EColv$\{v_i, v_j\}$": EColv$\{v_i, v_j\}$= $\cup_k\{Pred_k\}$ / $\forall w: Pred_k(w) = False$; where $Pred_k$ is a predicate defined using two or many variables and computed on different system states. These predicates define the collision set between these variables' values over the scenario execution w (i.e., the relations that must not exist between these variables over w). For example, $EColv\{v_3, v_4\} = "v_4(s+1) = v_3(s)"$, means that from one state to another, the value of v_4 should not be equal to the value of the variable v_3 in the previous system state.

5 Scenarios generation

In this section, the method used to generate actions and the method proposed to move from global executable scenarios to local ones will be presented.

5.1 Actions generation

An action can be defined as a relation between two system states s_i and s_{i+1}: A(s_i) = s_{i+1}[Rekhis and Boudriga, 2005a] [Rekhis and Boudriga, 2005b]. Indeed, by each action execution, the system moves from one state to another starting from an initial secure state s_0. In other words, an action A causes the modification of the values of a set of variables $V'(V' \in Var)$, which can be defined as the simultaneous execution of a set of functions f_{v_i}. Consequently, an action A can be modeled as follows:

$$A = \wedge_i f_{v_i} \; where \; f_{v_i} \in AtAc$$

Thus, any event occurred in the system will be modeled as a set of atomic actions that are executed simultaneously modifying the values of system variables and ensuring the generation of successive system states.

To clarify the usefulness of the SysCons library, we present the following example: Consider a system modeled using the set of variables $Var = \{v_1, v_2\}$, and two functions $f_{v_1}(s)$: $||v_1||_{s+1} \in \{0,1,2\} \; \forall \; ||v_1||_s$ and $f_{v_2}(s)$: $||v_2||_{s+1} \in \{0,1\} \; \forall \; ||v_2||_s$.

The SysCons library contains two conditions:$Scolv\{v_1, v_2\} = \{||v_2|| = 1 \; iff \; ||v_1|| = 1\}$ and $Ecolv\{v_1, v_2\} = \{ \; if \; ||v_1||s = 2 \; then \; ||v_2||_{s+1} = 1\}$. Starting from the predefined initial system state s_0 =< 0, 0 > (i.e., v_1= 0 and v_2 = 0), six next possible states could be generated (we apply the functions f_{v_i} for each variablev_i and generate all the possible combinations) s_{11}=< 0, 0 >, s_{12} =< 0, 1>, s_{13} =< 1, 0 >, s_{14} =< 1, 1 >, s_{15} =< 2, 0 >, s_{16}=< 2, 1>. Applying the conditions defined in the SysCons library, the states s_{12}=< 0, 1> and s_{16} =< 2, 1> will be removed as based on the first condition v_2=1 if and only if $v_1 = 1$. The next states generated from s_{15} =< 2, 0> will be s_{21} =< 0, 0>, s_{22}=< 0, 1>, s_{23} =< 1, 0>, s_{24}=<1, 1>, s_{25} =< 2, 0>,s_{26} = < 2, 1 >. The first condition in the SysCons library will serve to remove the states s_{22} =< 0, 1> and s_{26} =< 2, 1>, and the second condition will remove the states s_{21} =< 0, 0 >, s_{23} =< 1, 0 >, and s_{25} =< 2, 0 >. Consequently, only the state s_{24} =< 1, 1 > will be retained.

5.2 Global Executable Scenario

Starting from an initial safe system state $s_0$and by using the defined functions f_{v_i} in the AtAc library, the model will generate the set of possible next states $S_1 = \cup\{s_1\}$ that verify the system constraints defined in the library SysCons. Then, we generate the next states following any state in the set S_1. The process is repeated until we obtain a graph of scenarios execution. A scenario W in this set is modeled as W = $< s_0, A_0, s_1, \dots, s_n >$ where each state s_i is generated from a previous state s_{i-1} after executing action A_{i-1} and verifying that it satisfies the defined constraints. As the values of all the variables $v \in Var$ are considered in defining the system state s, we note this scenario by a Global Executable Scenario (GES) W = $< s_0, A_0, s_1, \dots A_{n-1}, s_n >$, $i \in \{1, .., n\}$ where n presents the number of actions of the scenario under execution W, a state s_i provides the valuation of all the system variables Var, and A_i is an action that once executed can modify one or many variable values in Var.

5.3 Local Executable Scenario

We developed in this model the concept of Local Executable Scenario (LES) to describe the execution of an attack scenario at one part of a distributed system. To obtain a local executable scenario, we divide the system variables Var into a set of groups based on a chosen feature that can be the security solution location or the nature of the supervised system resources. Based on this feature, the system can be divided into a set of layers L=$\{L_1, L_2, \dots, L_n\}$, where each layer L_i will be modeled by a set of local variables in Var. We define by a global executed action A_i, a set of subactions A_i^j executed simultaneously in an atomic manner. Thus, the global action $A_i$part of a scenario W can be modeled as: $A_i = A_i^1 | A_i^2 | \dots | A_i^K$, where K presents the number of layers decomposing the supervised system, and the local subaction A_i^j is part of the action A_i that impacts solely the values of the variables V_j. Hence, A_i^j will move the subsystem j from the substate s_i^j to the next substate s_{i+1}^j. Thus, the GES W will be decomposed into a set of LES W^j that we model as:

$$W^j =< s_0^j, A_0^j, s_1^j, \dots, A_n^j, s_n^j >$$

If the values of the variables belonging to the layer L_j stay unmodified after the execution of the action A_i, the subaction A_i^j will be replaced by the identity action Id. To clarify the concept of GES and LES, we provide the example presented in [Figure 1] where we divided the system into three layers according to the nature of the supervised system resources (physical resources, network resources and system software resources). As a result, the global executable scenario W=$< s_0, A_0, s_1, A_1, s_2, A_2, s_3 >$ is divided into three LES, say: $W^1 =< s_0^1, A0^1, s_1^1, A1^1, s_2^1, A2^1, s_3^1 >$ (the execution of W in the physical layer (L_1)), $W^2 =< s_0^2, A0^2, s_1^2, Id, s_2^2, A2^2, s_3^2 >$ (the execution of W in the network layer (L_2)) and $W^3 = < s_0^3, A0^3, s_1^3, A1^3, s_2^3, A2^3, s_3^3 >$ (the execution of W in the software layer (L_3)). The action A1 has no impact on the network layer and its execution does not modify the system substate s_1^2, thus the action $A1^2$was replaced by the action Id. Moreover, a global action is in reality a set of simultaneous executed subactions. For example, A0

is in fact, A0 = $A0^1|A0^2|A0^3$, where $A0^1$ presents the impact of A0 on L_1, $A0^2$ denotes the impact of A0 on L_2, and $A0^3$ presents the impact of A0 on L_3.

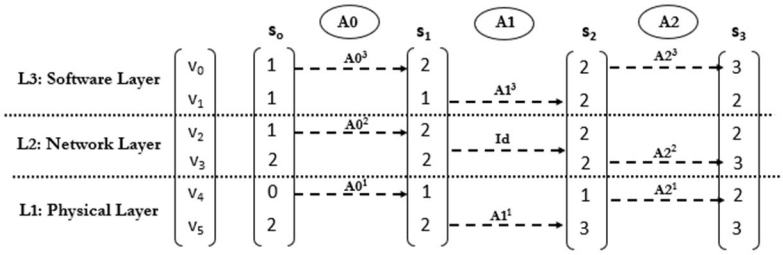

Figure 1: Local Executable Scenario

6 Security solution modeling

In this section, we will define and model the deployed security solutions (observer agents OA) and detail the used parameters.

6.1 Observer Agent role and configuration

In this work, an observer agent OA_i designates a security solution used to supervise a set of local variables V_j belonging to a specific system layer L_j that we model as OA(L, DD, $obs()$, Rdb), where:

L: Each OA is deployed at a specific system layer L which limits its ability to supervise all system parameters.

DD: The Defense Domain DD characterizes the variables $V \in Var$ belonging to a specific layer L that an OA is able to observe (i.e., it can determine their values over different system states).

Rdb: To each OA, a local reaction database Rdb is assigned modeling the countermeasures that it can execute.

$obs()$: The OA configuration is modeled by an observation function $obs()$describing when and how this OA observes the modification of the variables values that belong to its DD in each system state s. Indeed, $obs()$ is a function that associates to each variable v of the DD its observed value in a given system state s, that we denote by $||v||_s$. It is worth noting that in case where the value of a variable v becomes invisible, we model it as $||v||_s = \varepsilon$. A detailed description of this function is presented in the next subsection.

6.2 Observation function modeling

Each OA is characterized by an observation function $obs()$ applied to a set of assets that it monitors, generating different observation data. The observation of a variable $v \in Var$ under a predefined predicate $Pred_{SCdt}$ when the observation function $obs(Pred_{SCdt}, v)$ is used, is simply the variable valuation during the different observed system states whenever $Pred_{SCdt} = True$, and ε otherwise. In this work, we define the

predicate $Pred_{SCdt}$ to characterize the security conditions ensuring the proper functioning of the deployed OA (if these predicates are not satisfied, the solution becomes unable to correctly observe and monitor the supervised system). For example, if the transferred data are encrypted, the OA will be unable to observe the network and generate observations. The function $obs(Pred_{SCdt}, v)$ is defined as follows: $obs(Pred_{SCdt}, v)$=$[||v||_0, ||v||_1, \dots, ||v||_n]$ if $Pred_{SCdt} = True$. Based on the computed predicate $Pred_{SCdt}$, v can be non observable (it is non visible and its value cannot be interpreted directly), continuously observable (it is always visible and its value is recorded in each system state), discontinuously observable (it is visible, but its value is interpreted only if it changes from one state to another), indirectly observable (the observable value is a combination of v and at least another variable v0) or dynamically observable (the variable can be observed only if a predicate computed over the system state is true). A detailed description of the types of observation functions can be found in our work [Lakhdhar et al., 2017].

Taking the example presented in [Figure 2] where the system is modeled using the variables $Var = \{v_1, v_2, v_3\}$ and supervised by an observer agent OA. We define the predicate $Pred_{SCdt} \triangleq v2 \neq 1$. If we consider that v1 is continuously observable, v2 is discontinuously observable and v3 is non observable, the generated observations by OA are as follows: O_{v1} = [1, 1, 2, 3] , O_{v2} = [0, 1] and O_{v3} = [ ]. From state s_3, the OA will not generate any observation data as $Pred_{SCdt} = False$.

	s_0	s_1	s_2	s_3	s_4
v1	1	1	2	3	3
v2	0	0	0	1	1
v3	0	0	0	0	1

Figure 2: Observation Functions Example

6.3 Reactions database Rdb

In the proposed model, a local reaction database Rdb will be assigned to each deployed OA defining the countermeasures that it can execute once a malicious event is detected. In each Rdb a set of reactions R is defined where each reaction is modeled by three parameters R(Pred, Act, Alert):

- Pred: A predicate to be computed over the collected observations O generated by the observer agent OA.
- Act: The action that the OA will execute once Pred = True.
- Alert: An alert Alt(O) will be generated in conjunction with the executed action Act.

In our model, different actions can be triggered by an OA. As an example, we cite the following ones:

Block: Based on the collected observation O, the OA will block the scenario execution if R.Pred(O) = True.

Reconf(obs()): Based on the collected observation O, if R.Pred(O) = True, Reconf(obs()) will be executed and the observation functions obs() of the OA will be reconfigured to improve the system observability. For example, some variables are invisible unless a threshold is reached, the reconfiguration of the observer modifies that threshold.
Activate (Agent): Based on the collected observation O, if R.Pred(O) = True, Activate(Agent) will be executed, so that the asleep OA designated in the parameter "Agent" will be activated. In fact, we suppose that a set of OA is initially deactivated to reduce resources overhead. Thereby, Activate(Agent) will enhance the system observability. Indeed, if an OA is unable to decide whether the generated observations are related to malicious or legitimate action and whether a reaction has to be triggered, activating new agents will help it in getting more complete information about what is occurring in the system.
Activate (R): Based on the collected observation O, if R.Pred(O) = True, Activate(R) will be executed, and therefore deactivated rules in the database will be activated and used in the remaining part of the scenario execution. Actually, we suppose that in each Rdb, a set of reactions is initially deactivated to reduce the processing time and complexity.
Deactivate(R): Based on the collected observation O, if R.Pred(O) = True, Deactivate(R) will be executed, and a set of rules defined in the OA database will be deactivated as they become useless or they are replaced by new activated ones to keep the processing time and complexity reduced.

To manage the reactions execution, we define the following rule: In each Rdb, the reaction rules should be ordered, in such a way, the first enabled reaction where its predicate is satisfied (Pred(R) = True) is the one that will be executed and the subsequent ones will be ignored.

7 Attack contextualization

In this section, we will define the concept of context, detail the attack contextualization process and define the concept of observed local scenarios.

7.1 Context definition

In each cyber system and based on the deployed security solutions and their configurations, and the adopted security rules, the intruder may or may not damage the supervised system. Consequently, introducing the concept of attack execution context is highly important to well defend the cyber system. In our proposed model, we define the context C as the set of deployed security solutions and their configurations, that we model as a set of tuple information $C =< (OA_1, L_1, DD_1, obs_1, Rdb_1), \ldots, (OA_n, L_n, DD_n, obs_n, Rdb_n) >$, where each tuple represents a deployed security solution and its configuration.

7.2 Observed Local Scenario

As we mentioned previously, a GES is in reality a set of LES. But, the latter will not be totally visible. In fact, based on the OA' observation functions and DD, the LES can be

completely observed by some of them, partially observed by other ones and totally invisible by the rest. Furthermore, the deployed OA can trigger different security reactions in each step of the scenario execution. These triggered reactions have an impact on the scenario execution, which needs to be considered to model the actual executable scenario. In this context, we introduce the concept of Observed Local Scenario (OLS) which describes in addition to the observed part of the generated LES, the OA' behavior and reaction to each execution step. Thus, the alerts list will be updated each time an alert is generated, the activated agents and/or rules will contribute in detecting and reacting to the next execution steps until a blocking reaction is executed. Consequently, given a scenario S =$<s_0, A_0, s_1, \dots, s_n>$, a supervised system divided into a set of layers L and a network of OA. The scenario contextualization consists in executing three main steps: a) executing S in the supervised system; b) each OA_j will generate a set of observation O; and c) check the predicates defined in the OA' Rdb and once the predicate of block reaction is equal to True, all the remaining states of the scenario under execution will be deleted as the proposed system will remove all the states $<s_{i+1}, \dots, s_n>$ if $R.Pred_{s_i}(O) = True\ and\ R = Block()$.

8 Context-based cyber defense methodology

In this section, we detail the proposed approach and discuss its characteristics.

8.1 Methodology description

To implement the proposed cyber defense model, we start by defining the set of system variables Var, and the libraries AtAc and SysCons. After that, the system will be divided into a set of layers L= $\{L_1, \dots, L_n\}$based on a chosen feature and the set of parameters $V_i \in Var$ modeling each layer L_i have to be fixed. Subsequently, we configure the set of deployed OA.

Our methodology is presented in Algorithm1. We start with the first secure system state s_0. Then, we generate the set of all subsequent states using the library AtAc (we apply the function f_{v_i} for each variable v_i and generate all the possible combinations). From the generated states, we eliminate the ones where the constraints on the SysCons library are not satisfied. Based on the layer that it supervises, each OA will generate a set of observation data. At this point, the reaction database Rdb of the OA will be checked and the first enabled reaction (if it exists) will be triggered. From the remaining states, a set of next ones will be generated using the same approach leading to the generation of all possible scenarios where the security incidents are identified by reaching a state where a violation of safety property is detected. The latter is defined as a predicate to be computed over the generated observations in each system state. This process will be repeated until a block reaction is executed by one OA, or the scenarios under executions reach an unsafe state. The proposed approach is exemplified with the flowchart presented in [Figure [fig:Approach-Flow-Chart]].

Algorithm 1 Detailed Description of our Context-based Scenario Defense Methodology

1. Initialization: OAlist{}: list of activated OA, R_{init}: initial activated reactions in the Rdb
2. SysEnd = False: a predicate that becomes True once the scenario execution is blocked
3. k = 0
4. Repeat until (SysEnd = True)
(a) Let s_k be the initial state of S
(b) Generate $S = \bigcup\{s_{k+1}\}$ using AtAc library
(c) Let $S' = S \setminus \{s_{k+1} / \ s_{k+1} \nvDash SysCons\}$
(d) For each $s_{k+1} \in S'$
(e) For each system layers L_i
    i.Compute obs(s_k^i)
        A. For each OA
            - Compute the set of enabled reactions R'={R/Pred(R) = True}
            - If R'≠0, Execute the first enabled reaction (Head(R'))
            - If Act = Block then SysEnd = True;
            - If Act = Reconf(obs()) then, Reconf(OA.obs()) for the considered OA;
            - If Act = Activate(Agent) then OAlist = {OAlist, Agent};
            - If Act = Activate(R) then Rdb = {R_{init},R};
            - If Act = Deactivate(R) then Rdb = {R_{init} \ R};
        A. End for
    ii. End for
5. $s_k \leftarrow s_{k+1}$

6. End

In contrast to state transitions graphs, our approach is characterized by: a) the proposed graph is an observation-based one as on each scenario of the generated graph a set of observations will be produced. These observations are computed over the past executed steps and used to generate future ones; and b) The actions are dynamically generated as for each state s_i a set of next possible states s_{i+1} will be generated. The latter depends on the values taken by the variables in state s_i, and will define the next values based on the deployed AtAc and SysCons libraries. Accordingly, in the same level of the generated graph, s_i and s_j will generate different next possible states s_{i+1} and s_{j+1}.

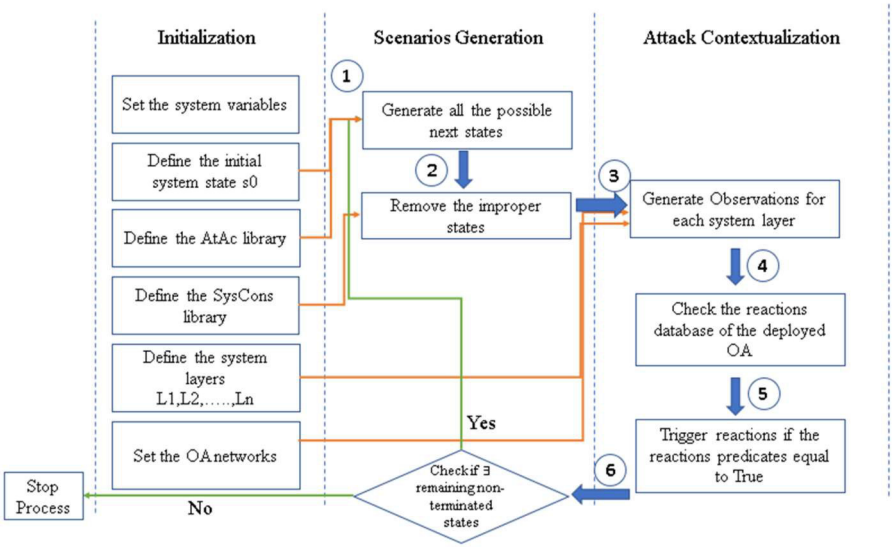

Figure 3: Context-based Cyber Defense Methodology Flowchart

8.2 Discussion

In this work, we presented a context-based defense model for assessing cyber systems' ability to defend against known and unknown attack scenarios. The proposed model can be discussed with respect to three major points:

The generation of unknown attack scenarios: the model we developed did not refer to any pre-built attack scenarios database. Rather, the actions are generated progressively based on the system specifications. In fact, defining the system variables and presenting an atomic action library AtAc describing the variable values modification from one state to another, and using SysCons library to remove all the improper states, all possible known and unknown scenarios will be generated in conformance with a defined attack context.

Attack contextualization: The attack execution context was introduced and modeled as the set of deployed observer agents and their configurations. In fact, in each step of a scenario execution, the deployed solutions can trigger (based on their configurations) different security reactions which can either block the scenario or allow it. This concept was presented in our work as the observed local scenario which allow us to perform a step-by-step assessment of the system resilience to attack executions under a predefined context.

Graph parameters: In the proposed model, from each state, all the possible next states will be generated to construct attack scenarios. The generated graph depth depends on four parameters which are: the system variables, the AtAc library definition, the conditions defined in the SysCons library and the predefined context. First, it is needed to accurately choose the system variables. Also, the functions in the AtAc library need to be well defined so that the generated states will have a finite number.

Furthermore, the conditions in the SysCons library need to be specified to remove all the non-logic states and thus reduce the graph breadth. Finally, stopping conditions need to be defined to ensure the graph termination. In fact, a scenario under generation is terminated once the deployed OA block its execution (based on the considered context) or a safety property is violated. We also consider the use of a set of heuristics to terminate infinite scenarios. Examples of these heuristics, include, but are not limited to: a) Define a priori maximum depth of a generated graph N. Thus, if a scenario under generation did not reach an unsafe state after N states, it can be terminated; b) Avoid repeating the execution of the same action inside a scenario more than a predefined number of times, says Thresh. Thus, if in the same scenario an action A_i is repeated more than Thresh times, the scenario generation can be terminated; and c) Avoid continuing the generation of a scenario if every system variable has taken all the possible values (defined in its domain) though the different previous system states in the same scenario.

9 Performance evaluation

In this section, we present the results of the implementation of our proposed approach using python applied to the same attack that we studied in our previous work which is the WannaCry attack [Lakhdhar et al., 2018a]. The latter is a worldwide ransomware attack that occurred on May 2017 and affected more than 300000 computers across the world [Chen and Bridges, 2017]. In fact, we aim to prove that using our approach, the WannaCry attack, in addition to other attacks (known and unknown), can be detected if they target the supervised system. First, we need to choose the system variables. In our system, we find three sets of variables: a) variables that can be remotely supervised using monitoring systems; b) variables related to network resources that can be supervised locally by deploying network IDS; and c) variables related to system application monitoring that are locally supervisable using host IDS. Consequently, the supervised system will be divided to three layers as follows:

Local variables related to system applications: To monitor this layer, we define four parameters:

- v_1: A boolean variable that supervises the file access control policy, which is initially equal to 0 and changes to 1 once a violation of at least one of the file access control policy is detected.
- v_2: Represents the number of available critical services among the DB-service, and the email-service. Initially $v_2 = 2$ as the two services are available. It changes to 1 if one of them is terminated or cannot run normally, and it becomes equal to 0 if all the services become unavailable.
- v_3: A boolean variable that is initially equal to 0 and changes to 1 once the installation of a suspect program is detected.
- v_4: Supervises the user gained privileges. v_4= 0 if no access is gained, v_4 = 1 when a user access is gained and it changes to 2 once a root access is gained.

Local variables related to network resources: This layer will be modeled using two parameters:

- v_5: Monitors the network traffic volume. It is configured to take one of three possible values $\{t_1, t_2, t_3\}$. $v_5 = t_1$ if the network traffic volume is within its

average values, its value changes to t_2 if its rate increases tolerably, and it becomes t_3 once its rate grows in an unusual manner.

- v_6: Monitors any unauthorized connection to a suspicious system. $v_6 = LN$ if the destination IP addresses are within the local network and $v_6 = EN$ if the destination IP addresses are from an external network.

Remotely supervised variables: This layer is modeled using one parameter:

- v_7: A boolean variable that supervises the system performance. It takes 0 if the system functions properly and, its value changes to 1 once a degradation in the system performance is detected.

To generate all the possible scenarios, we start by initializing the AtAc library as follows:

- $f_{v_1}(s)$: $||v_1||_{s+1} \in \{0,1\}$ if $||v_1||_s = 0$ else $||v_1||_{s+1} = 1$
- $f_{v_2}(s)$: $||v_2||_{s+1} \in \{0,1\}$ if $||v_2||_s = 0$ else if $||v_2||_s = 1$, $||v_2||_{s+1} \in \{0,1,2\}$ else $||v_2||_{s+1} \in \{1,2\}$
- $f_{v_3}(s)$: $||v_3||_{s+1} \in \{0,1\}$ if $||v_3||_s = 0$ else $||v_3||_{s+1} = 1$
- $f_{v_4}(s)$: $||v_4||_{s+1} \in \{0,1,2\} \forall ||v_4||_s$
- $f_{v_5}(s)$: $||v_5||_{s+1} \in \{t1, t2\}$ if $||v_5||_s = t1$ else if $||v_5||_s = t2$, $||v_5||_{s+1} \in \{t1, t2, t3\}$ else $||v_5||_{s+1} \in \{t2, t3\}$
- $f_{v_6}(s)$: $||v_6||_{s+1} \in \{LN, EN\} \forall ||v_6||_s$
- $f_{v_7}(s)$: $||v_7||_{s+1} = \{\left||v_7|\right|_s + 1 \; if \; ||v_7||_s = 0$ else $||v_7||_s - 1\}$

If we consider the initial system state s_0 =(0, 2, 0, 0, t1, LN, 0) and by referring to the AtAc library, our system can reach 192 possible next states from s_0. However, many of them are incoherent since we have not already used the constraints library. Consequently, based on the system variables, we define a set of constraints in the SysCons library as follows:

$Scolv\{v_1, v_4) = \{\left||v_1|\right| = 1 \; iff \; ||v_4|| = 2\}$: This rule indicates that the access control policies are restricted to users with root privileges. Thus, if an attacker wants to change them, he needs to gain a root access to the victim system.

$Scolv\{v_3, v_4) = \{\left||v_3|\right| = 1 \; iff \; ||v_4|| \neq 0\}$: This constraint indicates that to install any program, the attacker needs to have at least a user access privilege.

$Scolv\{v_5, v_7) = \{\left||v_5|\right| = t3 \; iff \; ||v_7|| = 1\}$: This constraint indicates that once the network traffic throughput reaches t3, the system performance is certainly degraded.

The application of these constraints reduces the generated set of next states from 192 to 112 states. This means that the system reduces the number of possible generated states by more than 40%. To show the importance of the used system constraints to logically reduce the number of the generated states, we make a comparison between the number of generated states with and without these constraints, considering a graph of depth 3 (3 states at maximum for every scenario). The results are presented in table1. The same should hold for scenarios with larger depth.

	1st graph level	2nd graph level	3rd graph level
Number of generated states without applying system constraints	192	32,400	3,981,312
Number of generated states after applying system constraints	112	10,800	808,704

By analyzing the set of scenarios that can be generated, we find the description of the WannaCry attack as shown in [Figure 4]. The scenario is composed of nine states. The first action $A_0 = \Lambda\{f_{v_5}\}$ represents the scanning action carried out by the intruder over port 445, $A_1 = \Lambda\{f_{v_5}, f_{v_7}\}$ represents the scanning action and the system performance degradation, and so on until the last action which consists in terminating the running services to start encrypting the victim data $A_8 = \Lambda\{f_{v_2}\}$. To study the resilience of the supervised system to various attack executions under a predefined context, we are going to deploy a network of OA to supervise system parameters. In this case study, three OA were deployed and configured as follows:

OA1: (S, $\{v_2, v_4\}, obs_1(), Rdb_1$); $obs_1()$ = {obs(v_2, s) =$||v_2||_s$ if $v_2 \geq 0$ else ε; obs(v_4,s) = $||v_4||_s$ if $v_4 \geq 0$ else ε} and Rdb_1={$(v_2 \neq 2, Block)$}: Denotes an IPS that supervises the connections' number for each service and once the latter exceeds a predefined threshold, it will block them.

OA2: (N, $\{v_5\}, obs_2(), Rdb_2$); $obs_2()$ = {obs(v_5, s) =$||v_5||_s$ if $v_5 \geq t1$, else ε} and $Rdb_2 = \{\}$: Denotes a NIDS supervising the network traffic.

OA3: (Ph, $\{v_7\}, obs_3(), Rdb_3$); $obs_3()$ = {obs(v_7, s) =$||v_7||_s$ if $v_7 \geq 0$, else ε} and $Rdb_3 = \{\}$: Represents a monitoring system that supervises the system performance by periodically sending test packets and checking the response.

Figure 4: Snapshot of Graph presenting WannaCry Attack

Based on their observations, the deployed OA blocked the states where $v_2 \neq 2$, reducing the generated states from the initial state to only 56 states. For the WannaCry attack, the fragment left of its scenario is presented in [Figure 5].

In this stage, we are going to change the context under which the model generates the possible scenarios and study its eect. The new context will be:

C{{OA1,OA2,OA3},{$obs_1(\,)$, $obs_2(\,)$, $obs_3(\,)$}, {Rdb_1, Rdb_2, Rdb_3}; $obs_1(\,)$ = {obs(v_2) =$||v_2||_s$ if $v_2 \geq 0$, else ε}; $obs\,(v_4\,)$ = $||v_4||_s$ if $v_4 \geq 0$, else ε; $obs\,(v_3\,)$ = $||v_3||_s$ if $v_3 \geq 0$, else ε} and $Rdb_1 = \{(v_2 \neq 2, Block);\ (v_3 = 1, Block); Rdb_2 = \{\};$ $Rdb_3 = \{\}$. In fact, we added the variable v_3 to OA1' DD and we activate a rule in the OA1' Rdb. After modifying the context, we re-execute the attack scenarios generation process. Under the new context, our model generates only 32 possible states starting from the initial state s_0 and the WannaCry attack scenario was blocked from the state s_5.

Despite the WannaCry, the proposed approach succeed to generate various attack scenarios such as the recent released "Shamoon V3" attack that occurred on the 10th December 2018 [2]. Shamoon attack: W=<(0, 2, 0, 0, t1, LN, 0), GainAccess, (0, 2, 0, 1, t1, LN, 0), InstallPgrm, (0, 2, 1, 1, t1, LN, 0), ConnectC2, (0, 2, 1, 1, t1, EN, 0), DownDropper, (0, 2, 1, 1, t2, EN, 0), ElevatePriv, (0, 2, 1, 2, t1, LN, 0), ScanNetwk, (0, 2, 1, 2, t2, LN, 0), ScanNetwk, (0, 2, 1, 2, t3, LN, 1), RunWiper, (0, 2, 1, 2, t2, LN, 1)>. By re-executing the attack scenarios generation process under the following modified context:

C{{OA1,OA2,OA3},{$obs_1(\,)$, $obs_2(\,)$, $obs_3(\,)$}, {Rdb_1, Rdb_2, Rdb_3}; $obs_1(\,)$ = {obs(v_2) =$||v_2||_s$ if $v_2 \geq 0$, else ε}; $obs\,(v_4\,)$ = $||v_4||_s$ if $v_4 \geq 0$, else ε; $obs\,(v_3\,)$ = $||v_3||_s$ if $v_3 \geq 0$, else ε}, $obs_{\,2}\,(\,)$ = $obs\,(v_5\,)$ =$||v_5||_s$ if $v_5 \geq 0$, else ε; $obs\,(v_6\,)$ =$||v_6||_s$ if $v_6 \neq \{\}$ else ε }and $Rdb_1 = \{(v_2 \neq 2, Block);\ (v_3 = 1, Block);\ Rdb_2 =$ $\{\ v_6 = EN, Block\}$; $Rdb_3 = \{\}$, the Shamoon attack was blocked from the state where the malware tried to connect to C2 server to download the malicious payload.

10 Conclusion

In this work, we developed a context-based defense model for assessing the cyber system's ability to defend against known and unknown attack scenarios. We developed a complex model allowing the generation of both known and unknown attack scenarios from system modeling. In addition, a distributed observer agents' network was deployed to supervise different system parameters. In this work, we also developed the attack contextualization concept, and proved its importance in assessing the system resilience to attack executions. We presented the concept of observed local scenario allowing the step-by-step observation of a scenario execution, the assessment of observer agent reaction and the identification of the step at which the scenario can be blocked.

[2] McAfee Report, https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/shamoon-returns-to-wipe-systems-in-middle-east-europe/

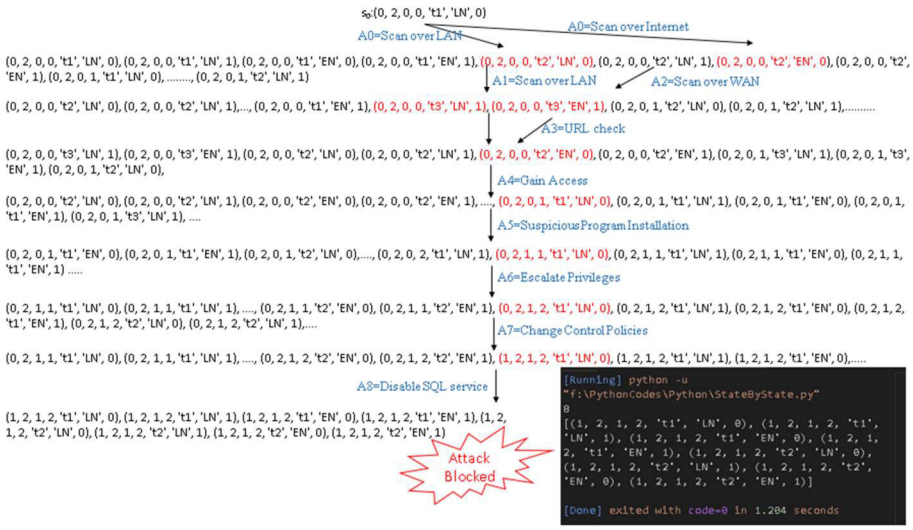

Figure 5: Possible scenarios after deploying OA

References

[Aleroud and Karabatis, 2017] Aleroud, A. and Karabatis, G. (2017). Contextual information fusion for intrusion detection: a survey and taxonomy. *Knowledge and Information Systems*, 52(3):563-619.

[AlEroud and Karabatis, 2017] AlEroud, A. and Karabatis, G. (2017). Methods and techniques to identify security incidents using domain knowledge and contextual information. In *IFIP/IEEE Symposium on Integrated Network and Service Management*, pages 1040-1045.

[Aparicio-Navarro et al., 2017a] Aparicio-Navarro, F. J., Chambers, J. A., Kyriakopoulos, K., Gong, Y., and Parish, D. (2017a). Using the pattern-of-life in networks to improve the e_ectiveness of intrusion detection systems. In *IEEE International Conference on Communications (ICC)*.

[Aparicio-Navarro et al., 2017b] Aparicio-Navarro, F. J., Kyriakopoulos, K. G., Gong, Y., Parish, D. J., and Chambers, J. A. (2017b). Using pattern-of-life as contextual information for anomaly-based intrusion detection systems. *IEEE Access*, 5:22177-22193.

[Berenjian et al., 2016] Berenjian, S., Shajari, M., Farshidy, N., and Hatamianz, M. (2016). Intelligent automated intrusion response system based on fuzzy decision making and risk assessment. In *IEEE 8th International Conference on Intelligent Systems*.

[Chen and Bridges, 2017] Chen, Q. and Bridges, R. A. (2017). Automated behavioral analysis of malware a case study of wannacry ransomware. *arXiv preprint arXiv:1709.08753*.

[Ghorbani et al., 2016] Ghorbani, M., Ghorbani, H. R., and Hashemi, M. R. (2016). Configuration strategies for collaborative ids using game theory. In *24th Iranian Conference on Electrical Engineering (ICEE)*, pages 261-266.

[Giura and Wang, 2012] Giura, P. and Wang, W. (2012). A context-based detection framework for advanced persistent threats. In *International Conference on Cyber Security*, pages 69-74.

[Guezguez et al., 2017] Guezguez, M. J., Rekhis, S., and Boudriga, N. (2017). Observation-based detection of femtocell attacks in wireless mobile networks. In *Proceedings of the Symposium on Applied Computing, Morocco*.

[Hina and Dominic, 2016] Hina, S. and Dominic, D. D. (2016). Information security policies: Investigation of compliance in universities. In *3rd International Conference On Computer And Information Sciences (ICCOINS)*.

[Joshi et al., 2018] Joshi, C., Singh, U. K., and Kanellopoulos, D. (2018). An enhanced framework for identification and risks assessment of zero-day vulnerabilities. *International Journal of Applied Engineering Research,* 13(12):10861-10870.

[Keramati, 2016] Keramati, M. (2016). An attack graph based procedure for risk estimation of zero-day attacks. In *8th International Symposium on Telecommunications*.

[Kotenko and Chechulin, 2012] Kotenko, I. and Chechulin, A. (2012). Attack modeling and security evaluation in siem systems. *International Transactions on Systems Science and Applications*, 8:129-147.

[Lakhdhar et al., 2016] Lakhdhar, Y., Rekhis, S., and Boudriga, N. (2016). An approach to a graph-based active cyber defense model. In *The 14th International Conference on Advances in Mobile Computing & Multimedia (MoMM2016)*, Singapore.

[Lakhdhar et al., 2017] Lakhdhar, Y., Rekhis, S., and Boudriga, N. (2017). Proactive damage assessment of cyber attacks using mobile observer agents. In *The 15th International Conference on Advances in Mobile Computing & Multimedia (MoMM2017)*, Salzburg, Austria.

[Lakhdhar et al., 2018a] Lakhdhar, Y., Rekhis, S., and Boudriga, N. (2018a). A context-based model for validating the ability of cyber systems to defend against attacks. In *The Fourth International Symposium on Ubiquitous Networking*.

[Lakhdhar et al., 2018b] Lakhdhar, Y., Rekhis, S., and Boudriga, N. (2018b). Proactive security for safety and sustainability of mission critical systems. *IEEE Transactions on Sustainable Computing.*

[Li et al., 2017] Li, F., Xiong, F., Li, C., Yin, L., Shi, G., and Tian, B. (2017). Sram: A state-aware risk assessment model for intrusion response. In *IEEE Second International Conference on Data Science in Cyberspace*.

[Plosz et al., 2017] Plosz, S., Schmittner, C., and Varga, P. (2017). Combining safety & security analysis for industrial collaborative automation systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 187-198.

[Razzaq et al., 2009] Razzaq, A., Hur, A., Ahmad, F., and Haider, N. (2009). Multi-layered defense against web application attacks. In *The Sixth International Conference on Information Technology: New Generations*, pages 492-497.

[Razzaq et al., 2013] Razzaq, A., Hur, A., Ahmad, H. F., and Masood, M. (2013). Cyber security: Threats, reasons, challenges, methodologies and state of the art solutions for industrial applications. In I*EEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, Mexico.

[Rekhis and Boudriga, 2005a] Rekhis, S. and Boudriga, N. (2005a). A formal logic-based language and an automated veri_cation tool for computer forensic investigaion. In 2005 *ACM symposium on Applied computing,*, pages 287-291.

[Rekhis and Boudriga, 2005b] Rekhis, S. and Boudriga, N. (2005b). A temporal logic-based model for forensic investigation in networked system security. In *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*.

[Si-chao and Yuan, 2016] Si-chao, L. and Yuan, L. (2016). Network security risk assessment method based on hmm and attack graph model. In *17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*.

[Snidaroa et al., 2015] Snidaroa, L., Garciab, J., and Llinasc, J. (2015). Context-based information fusion: a survey and discussion. *Information Fusion*, 25:16-31.

[Vasilomanolakis et al., 2015a] Vasilomanolakis, E., Karuppayah, S., Kikiras, P., and Muhlhauser, M. (2015a). A honeypot-driven cyber incident monitor: lessons learned and steps ahead. In *8th International Conference on Security of Information and Networks*, pages 158-164.

[Vasilomanolakis et al., 2015b] Vasilomanolakis, E., Krugl, M., Corderoand, C. G., Muhlhauser, M., and Fischer, M. (2015b). Skipmon: A locality-aware collaborative intrusion detection system. In *IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, pages 1-8.

[Wan, 2009] Wan, K. (2009). A brief history of context. *IJCSI International Journal of Computer Science*, 6(2).