Bio-inspired Security Analysis for IoT Scenarios

Vincenzo Conti*

Faculty of Engineering and Architecture,
University of Enna KORE, Enna, Italy
E-mail: vincenzo.conti@unikore.it
*Corresponding author

Andrea Ziggiotto, Mauro Migliardi

Department of Information Engineering,
University of Padua, Padua, Italy
E-mail: andrea.ziggiotto@studenti.unipd.it, mauro.migliardi@unipd.it

Salvatore Vitabile

Department of Biomedicine, Neuroscience and Advanced Diagnostics,
University of Palermo, Palermo, Italy
E-mail: salvatore.vitabile@unipa.it

Abstract: Computer security has recently become more and more important as the world economy dependency from data has kept growing. The complexity of the systems that need to be kept secure calls for new models capable of abstracting the interdependencies among heterogeneous components that cooperate at providing the desired service. A promising approach is attack graph analysis, however, the manual analysis of attack graphs is tedious and error prone. In this paper we propose to apply the metabolic network model to attack graphs analysis, using three interacting bio-inspired algorithms: topological analysis, flux balance analysis, and extreme pathway analysis. A developed framework for graph building and simulations as well as an introductory to some IoT scenarios as use cases are also outlined.

Keywords: Attack Graphs; Network Security; Bio-inspired Techniques; IoT

Reference to this paper should be made as follows: V. Conti, A. Ziggiotto, M. Migliardi, S. Vitabile, 'Bio-inspired Security Analysis for IoT Scenarios', *International Journal of Embedded Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Vincenzo Conti is currently an Assistant Professor with the Faculty of Engineering and Architecture at the Kore University of Enna, Italy. He received his Laurea cum Laude degree and his Ph.D. in Computer Science from the University of Palermo in 2000 and 2005, respectively. His research interests include biometric recognition systems, programmable architectures and bio-inspired processing systems. He has joined the Editorial Board of many journal of the computer science area.

Andrea Ziggiotto is a master student in Computer Engineering at the University of Padua.

Mauro Migliardi got a Laurea in Electronic Engineering and a PhD in Computer Engineering from the University of Genoa. He is Associate Professor at the University of Padua and Adjunct Professor at the University of Genoa. He is in the Scientific Board of CIPI (IT) and ARC4DigiT (PT). He participated or lead research projects sponsored by the Italian Government, the EU and the US DoE. His main research interest is distributed systems engineering in general; recently he focused on mobile systems, cybersecurity, energy-aware security and IoT.

Salvatore Vitabile is an Associate Professor at the Department of Biomedicine, Neuroscience and Advanced Diagnostics, University of Palermo, Italy. In 2007, he was a Visiting Professor in the Department of Radiology, Ohio State University, Columbus, US. He is currently a member of the Board of Directors of SIREN (Italian Society of Neural Networks). His research interests include specialized architecture design and prototyping, bio-inspired systems, biometric authentication systems, and medical data processing and analysis..

1 Introduction

Over the years, computer security has acquired a central role in all the systems as the modern world has become more and more dependent on a seamless and timely flow of information and data. In gaining this central role, Computer Security has developed branches dealing with different aspects, from the most practical ones such as the one dedicated to the identification of vulnerabilities [1] and the development of automated protection techniques [2] and [3], to the more theoretical ones that strive to leverage mathematical methodologies to evaluate the security (or lack of) of systems [4], [5] and of the Cloud [6]. While the importance of the practical approaches is paramount and cannot be in any way denied, we claim that the growing complexity of the systems and the compositional nature of systems developed under the umbrella of the Internet of Things requires the development of modeling tools capable of capturing the characteristics and the behavior not just of the components but of the whole system as a network of interacting entities. To achieve this goal, in this paper we propose to exploit the metabolic modeling to complex computer-based systems. This formalism, with the appropriate mapping of security metrics onto the quantities typically involved in biochemical reactions, allows identifying critical paths and highlight spots that present the highest level of vulnerability for the system as a whole, hence, it is our opinion that it is well suited for the task of performing a holistic security analysis to complex systems. More in details, after describing the formalism we show an introductory use-case, namely the analysis of attack graphs where we translate the biochemical measurements involved in the metabolism of a cell into some typical security metrics; in this way, we can leverage the metabolic network to perform a security analysis.

Our use case will analyze a network, assuming that the vulnerabilities are already known. Starting from this information an attack graph will be created and then it will be analyzed by a tool that allows us to obtain values that determine which is the greatest risk and the minimum number of changes needed to morph the network into a new version that could be considered at least partially safe. In this paper we will take into account only traditional vulnerabilities while energy awareness as described in [7], [8], and [9] will be left to future work.

In the literature there are several techniques used for attack graph analysis. In [10], the attack graph is seen as a game to which the Stackelberg equilibrium is applied to search for the best strategy that defenders must apply to defend against attackers. Other applications are based on greedy algorithms, transforming the problem of finding the minimum set of critical attacks into the problem of Minimum hitting set. Through this technique it is possible to find a solution in a time that is linearly related to the size of the network. The graphs used in these articles are generated by the tool MulVAl. Many approaches, however, for the analysis of networks exploit the application of PageRank of Google [11] [12] on Büchi automata [13]. The approaches from which this work is inspired leverage attack graphs in which each node is associated with a probability. In [14] a methodology is proposed for the research of the minimum graph, which is the minimum path that allows the attacker to reach his goal. In [15], the attack graph is analyzed using the Monte Carlo method in order to obtain significant information on the economic value of the attack as well as on the risk of it. In [16] the network is studied using the probability theory. Similar works use exponential distributions to simulate the success of an attack and analyze network security using Markov models.

In this paper we propose a new approach that leverages the application of bio-inspired algorithms [17] to tackle the problem of attack graphs analysis. The paper envisions some new application scenarios for these algorithms that show a high level of adaptability to different fields of applications, as already seen in [18], [19]. The implemented and proposed framework is an extension of the BIAM software proposed in [18]. The framework allows the drawing and managing of complex networks and their analysis by means of bio-inspired algorithms derived from techniques of analysis originally designed for usage with metabolic networks. The framework also allows saving both the generated graph and the results obtained, hence it is possible to perform a "design, analyse, implement strengthening measures" loop over a candidate network. Because a network can be represented with a graph, the following article will use the two terms as synonyms.

The paper is organized as follows: in section 2 we describe the most common attack graphs; in section 3 we briefly introduce some of the security issues in modern networks; in section 4 we describe our bio-inspired framework; in section 5 we describe the IoT scenarios as case studies; and, finally, in section 7 we provide some concluding remarks.

2 Attack Graphs

Because of the value and importance that resources have within networks, they are increasingly subject to cyberattacks by hackers. The identification of vulnerabilities within any type of network is no longer as difficult as it used to be in the past, considering the rapid growth of current vulnerability scanning techniques. In most cases, cyberattacks are used to test and comprehend the vulnerabilities of a network. These very sophisticated attacks no longer use isolated vulnerabilities (i.e., vulnerabilities that are local to a system), on the contrary, multiple vulnerabilities of different networks are used and combined among them using causal relationships in order to damage the network. Attacks of this type, in the field of computer security, are defined as "multi-host/multi-stage" attacks; they are very advanced and they can be targeted

to any type of system behind firewalls, protected by multiple levels of security, or hidden inside a company network as they are able to bypass any type of software detection. A very useful tool for risk analysis used by network analysts, in order to understand what are the vulnerabilities of a network, but at the same time makes it possible to manage vulnerabilities following an IT attack is the attack graph. The attack graph is used in order to study the vulnerabilities of a network, both in a preventive and repressive way to an attack. Experts in computer security, whose main purpose is to provide information about the vulnerabilities of a system or network, use it in various areas of computer security, such as forensics:

- it is used by analysts to carry out analyses on the security of the network under examination and then find the vulnerabilities and correct them;
- it is used by cybercriminals to perform combined cyberattacks based on vulnerabilities of the network that the graphs attack detects.

For these uses, the nodes of an attack graph represent the attacker in a specific position within the network, while the arcs represent the actions taken by the attacker and the changes in the state of the network caused by such actions. The actions taken, for the most part, are exploits or exploit the vulnerabilities that the network, protocols or software have. There are different types and topologies of attack graphs, because each network is different from the others and has multiple or isolated high or low priority vulnerabilities [20]. In the following section, we describe some common types of attack graphs.

2.1 Attack Tree

Attack Trees [21] [22] are a prime example of attack graphs. Their success lies in the ability to provide information on how an attacker can achieve a goal in an intuitive way through simple logical operations of conjunction and disjunction between sub-activity. They show the logical connection between two events in an attack. Attack trees are usually composed by OR-gates and AND-gates. Events output from an OR-gate occurs only if one o more events of the input are present. Events output from an AND-gate occurs only if all events of the input are present. Attack Trees are a convenient way to show the ways in which a computer system can be attacked. They are represented as trees and there are different types of them. Figure 1 shows a first example, represented with a binary tree with sub-activities. If two nodes are connected by a conjunction operation, then the two sub-activities must both be performed to achieve the final goal. However, if the two nodes are connected by a disjunction node then at least one of the two targets must be achieved in order to achieve the final goal.

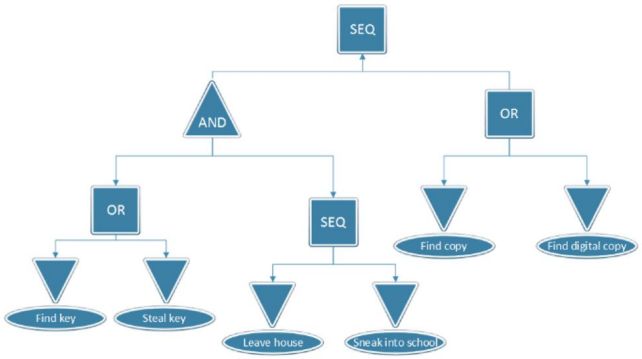


Figure 1 An example of Attack Tree

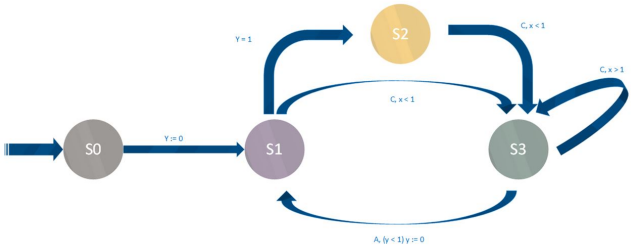


Figure 2 An example of Büchi Automata

2.2 Büchi Automata

The problem with Attack Tree is that it does not provide all the necessary and useful information, as opposed to methods that use stochastic timed automata or probabilistic stochastic automata. These methods use automata such as Büchi Automata that allow obtaining information on both the probability (input and output events) and the sequence of optimal operations. A Büchi automata is a non-deterministic finite automata working with infinite words represented by a tuple. For example, considering the following tuple: $Aut_B = (S_B, T_B, S_{B_0}, S_{B_a}, S_{B_f}, D_B)$

- S_B represents a finite set of states;
- $T_B \subseteq (S_B \text{x} S_B)$ is a transitional relationship;
- $S_{B_0} \subseteq S$ is a set of initial states;
- $S_{B_a} \subseteq S_B$ is a set of accepted initial states;
- S_{B_f} is a finite set of final state;
- D_B is a relationship from S_B to an atomic proposition.

Types of attack graphs based on Büchi automata are the "scenario graph", also called "Scenario Automata". The structure of these graphs depends on the model displayed. Scenario automata are cyclic directional graphs. A scenario is graph, cyclic with liveness property (Figure 2).

2.3 Advice model

Among the attack graph models is the advice model which helps to establish quantitative metrics of the security of a system. It is divided into three parts: attack execution graph, adversary profile and metric specification. An execution attack graph contains all the information about the system, the goals and the stages of attack. It is represented as a tuple:
$A_{t_{t_{Ex}}} = (A_{T_k}, a_d, K, S, G)$

- A_{T_K} represents the set of attack stages;
- a_d represents the access domain in the system;
- K knowledge items;
- S the set of attack skills;
- G goals.

Each of these elements is represented differently in the graph: Access domains are represented by green rectangles, the attack skills are represented by yellow rectangles, knowledge items are represented by blue ellipsis while the objectives are represented as red ellipsis. The attack phases are instead modelled from the following tuple: $a = (B_i, T_i, C_i, O_i, P_{r_i}, D_i, E_i)$

- B_i is a Boolean condition;
- T_i is the amount of time required to attempt the attack;
- C_i is the cost;
- O_i is a finite set of results;
- P_{r_i} assigns each result a probability to be obtained from the attack;
- D_i is the probability of an attack being detected if the condition of verification;
- E_i is the function that determines the next model of state after the condition has occurred

The attack stages operate on the model state, a tuple $s = (R_s, K_s, G_s)$, representatives, respectively, the access domain, the required knowledge, and the set of goals. The second part of the advice model is the adversary profile, which is represented by a tuple $(s_0, F, V, W_C, W_P, W_D, U_C, U_P, U_{n_D}, N)$ where

- S_0 is an initial state;
- F a function that attaches to the attack skill a real number between 0 and 1;
- V is the function that associates a positive value with each lens that represents the target value of the attack.
- the next three elements "W" represent the weights of costs, payoffs and detection probability

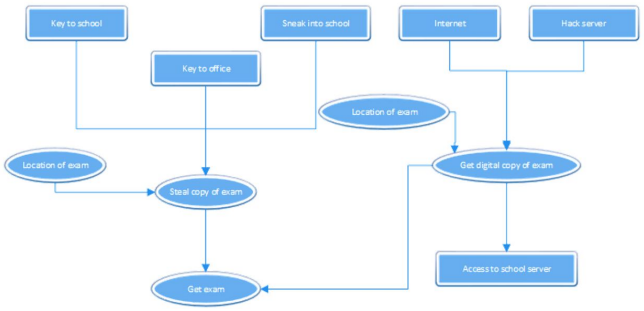


Figure 3 An example of Advice Model

- the next three elements "U" represent the function utility that maps the values of each attractiveness to a value between 0 and 1 called Utility scale;
- N is a planning horizon.

The third component is the metric specification that can be divided, in turn, into two types: State metrics and event metrics. While the first analyses the state model, the second analyzes the state changes, the attempted attack phases and their results. The Advice models then work through the same relationships that exist in Boolean algebra (Figure 3).

3 Security Assessment through Attack Graph

As actual computing systems are far too complex to be manually analyzed by a security analyst, there is a growing demand for techniques and tools that allow automatizing the security assessment of complex systems. At the basis of a successful automatic methodology there lie proper models enabling 1) to provide an abstract, simplified but complete view of the system under validation (SUV), and 2) to define strategies for the automatic assessment of the SUV. Current literature is plenty of works allowing to formalize the behavior of parallel systems, mobile applications [23], wired [24] [2] and wireless [25] [9] networks and so on. The primary challenge with such modeling activities is to prove the provided abstraction is general but complete enough to support a comprehensive analysis of the system. Regarding assessment strategies, these are expected to allow the definition of assessment/testing methodologies of the modeled SUV. The reliability of both models and assessment strategies depends on the number of vulnerabilities of the actual SUV that they allow unveiling. Among these, attack graphs provide a pure formalism to define the steps to attack an SUV to uncover vulnerabilities. Nowadays security analysts exploit attack graphs to keep track of the attack steps, manually. Albeit several approaches have been put forward in the literature to support a semi-automatic building of attack graphs, most of the proposals fail to provide reliable solutions. With the adjective reliable, we refer to the possibility to assess the security of

the SUV regarding both the time needed to carry out the analysis phase as well as the number of successful vulnerabilities that such analysis allows to discover. The current solutions for generating attack graphs provide only slight improvements of such metrics when compared to the manual building carried out by an expert security analyst. We argue that novel models should be put forward to try improving the reliability of the automatic generation of attack graphs. However, the mere building of SUV-specific graph attacks is a limited contribution, as each attack graph must be tested and evaluated on the SUV. Therefore, there is the need for comprehensive frameworks able to 1) generate attack graphs automatically, 2) automatically carry out security tests on the SUV by exploiting the generated attack graph, 3) evaluate the reliability of the attack. The previous considerations are valid for any type of SUV in computing systems, e.g., from concurrent processes to mobile environments. We argue that each category of SUV requires a specific solution for attack graph generation and validation. In this paper we put forward a solution based on bio-inspired algorithms for the definition and the building of attack graphs, and we specifically focus on LANs as SUV.

4 Framework Description: Characteristics and Functionalities

The proposed developed framework is based on three bio-inspired algorithms for network security analysis, namely: Topological analysis, Extreme Pathway analysis (Expa) and Flux Balance analysis (FBA). These three algorithms were designed to analyze metabolic networks, particular networks that describe the reactions that, in the case of cellular metabolism, lead to the production of nutrients for the cell. The topological analysis [26] allows obtaining information on the structure and topology of the network as: the number of vertex and edge, the hub nodes, the clustering and participation coefficients, and the diameter. The extreme pathway analysis [27], working on the stoichiometric matrix, allows obtaining all the paths from a source to a destination of the graph. The FBA [28] calculates the optimal path and the related optimal fluxes. However, this path must have a minimum input flux (called nutrient uptake) for cell survival. In the context of computer security, the metrics that are typical of the biochemical analysis performed in metabolic networks have to be mapped onto security parameters to translate the above mentioned types of analysis into security analysis. In this paper, we adopt the mapping that was first described in [29]. More in details, the weight of each arc of the network represents the risk that an exploit can cause a certain damage equal to the CVSS score of the exploit itself. The best path represents the path, which is the sequence of exploits, which maximizes the risk. This represents the sequence of exploits that maximizes both the damage suffered and the likelihood of suffering such damage.

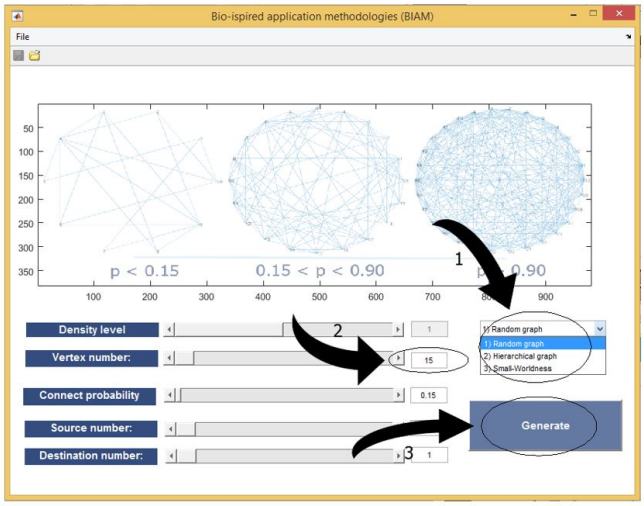


Figure 4 The task for the graph generation

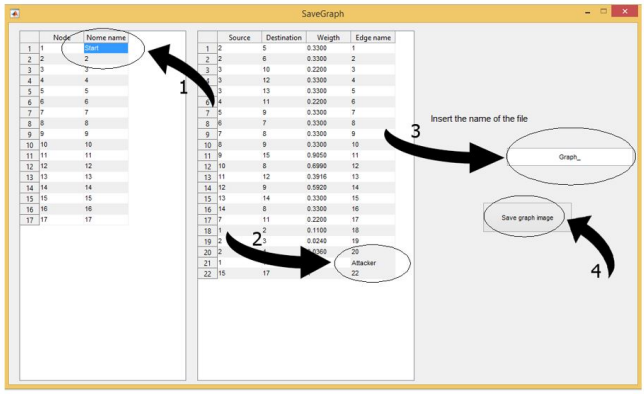


Figure 5 The task for the graph saving

The framework we describe in this paper, namely BIAM 2.0, is the enhanced version of the framework proposed by the same authors in [18]. The first version of BIAM framework was able only to generate scale-free networks and analyze them with bio-inspired methodologies. This new and enhanced version also allows building and handling random and small-world networks. Furthermore, BIAM 2.0 is equipped with a user-friendly GUI capable of showing in real time: the generated graph, the changes made and the results obtained from the analyses (see Figure 4).

By clicking the "Generate" button, it is possible to modify the generated graph before performing the analysis of the same. In addition, the framework also gives the option to save the generated graphs and then reload them later. Finally, once the analyses have been carried out, the framework allows saving the results on text files and also to save images of the graphs, with the possibility to assign a name to each node and link, to be able to draw up (see Figure 5).

Figure 1 shows the first window of BIAM 2.0, where a combo box allows selecting among three types of complex network; in this step it is possible to choose the desired graph and set the parameters for the random generation; then the graph can be generated by pressing the related

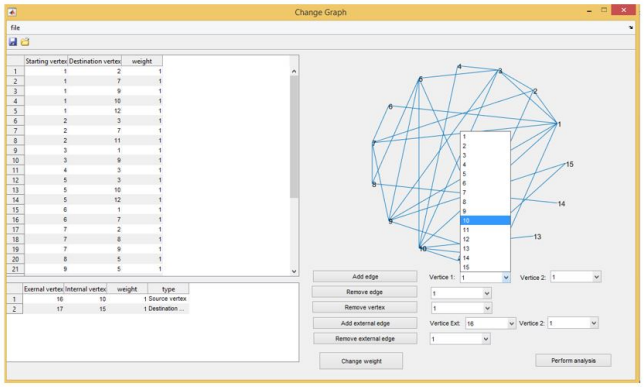


Figure 6 The task for the graph change

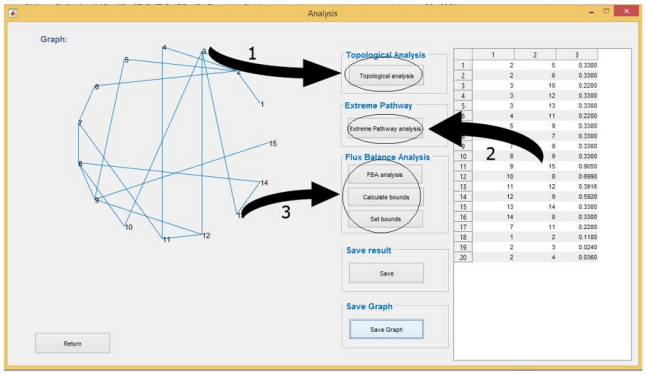


Figure 7 The sequence of task to perform the bio-inspired analysis

button. After this step has been performed, it is possible to modify the network/graph to be analyzed customizing it according to a model (see Figure 6).

Using the "Perform analysis" button, the framework (see Figure 4) shows the available types of analysis. It is possible to select to apply, to the generated network, each one of the three algorithms to single out the results of that specific analysis, or, using the "Save Result" button, all three algorithms can be simultaneously applied and the obtained results are saved into a text file. Figure 7 shows the sequence of buttons to be pressed to be able to perform the three algorithms individually.

Other implemented tasks are: modifying the bounds for the flux balance analysis, using the buttons "Calculate Bounds" and "Set bounds", and to save the network to be analyzed at a later time, giving the names to links and nodes, using the button "Save Graph".

5 Case Study

As described in section 2, attack graphs are special graphs that mathematically describe attack scenarios [20]. An example of an attack graph is the sequence of events and actions needed to carry out a terrorist act or a cyberattack. There are different types of attack graphs, each of which represents a particular piece of information about an attack. In various models, nodes can represent the target of the attacker or the phases of an attack. In general, nodes are associated with a cost, a cause or a probability of success. The choice between a model and another depends on which aspect of the network or which information is necessary to extract.

The attack graph model is realized on the arcs, the "exploits" are performed during the attack phase on the nodes of the states in which the network is located after the execution of each exploit. In this paper, the term "exploit" is used to represent all the actions that an attacker needs to perform to identify a vulnerability. The weight of each arc represents the risk factor of the corresponding exploit, calculated as the product between the damage suffered and the likelihood of suffering that damage. The variable damage acts with a random variable. For simplicity, it can be written as:

$$R = D * P \tag{1}$$

It represents the risk for a system to receive some damages. The probability of the exploit being performed is taken from the Common Vulnerability Scoring System (CVSS) score [30] of the same. The algorithm used to calculate the CVSS takes the name of the Common Vulnerability Scoring System and can be found on the NIST (National Institute of Standard and Technology) site [31]. The current version of the algorithm is version 3.0. The values of CVSS scores were obtained from the National Vulnerability Database (NVD) located on the same NIST site we have previously cited.

Each event has been considered independent from the previous and the analysis is considered independent from the attacker's knowledge. For the calculation of bounds in Flux Balance Analysis, two limit values have been considered, these two values correspond to the maximum risk (CVSS equal to its actual value) and to the minimum risk (CVSS equal to 1).

The algorithm used to calculate a CVSS from a metrics base and described in Figure 8 is based on a main equation derived from two sub-equations: the Exploitability Subscore equation and the Impact Subscore equation. While the Exploitability Subscore equation comes from the base metric exploitable metrics, the Impact Subscore equation is derived from the impact metric base. Typically, baseline and temporal metrics are specified by vulnerability analysts or vendors of security products or applications, because they generally have more accurate information. In contrast, environment metrics are specified by the end user of organizations because they know the impact information that has that vulnerability.

The CVSS score for each vulnerability has been derived from the NIST NVD looking for the vulnerability from its CVE ID. The reported case study is based on two fundamental hypotheses since everything is independent from the attacker's knowledge. It was considered that the attacker knows the right exploit for each vulnerability and that he is able to use it to his advantage. The first hypothesis is that the system

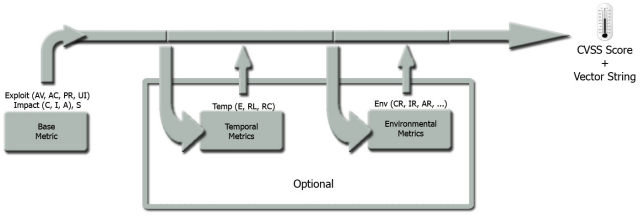


Figure 8 CVSS Score Calculation process

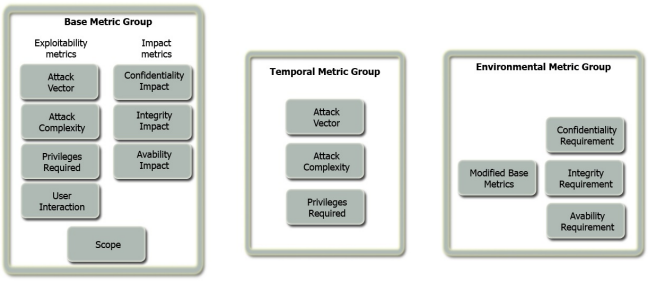


Figure 9 Parameters for the CVSS calculation

knowledge is very good and so on how the attack must be performed. The second hypothesis is that every exploit performed has no effect on successive events, so each event becomes independent from the previous ones (see Figure 9).

5.1 An IoT Scenario

For the creation of the scenario, at first, we thought to use only commercially available IoT devices and to build the graph using vulnerabilities related to each device, as collected from publicly available databases. This road, however, was not viable for two reasons: first, many of the investigated devices have still a very low degree of market penetration, hence information about the device security is quite limited; second, the architecture of IoT devices sold at present is strictly a vertical, cloud based silo, where every device talks almost exclusively with its own cloud and there is no significant level of communication between different devices. We argue that the future promised by IoT is a richly interconnected weave of devices, where information flows to generate added value services and to enhance the usefulness of single devices. Hence, we have decided to develop a scenario composed by devices, that although not currently present in the market, are realistic in the near future; subsequently, we assigned to these devices vulnerabilities taken from similar devices as described in databases such as the NVD or Exploit DB. Using these vulnerabilities we have built the attack graph.

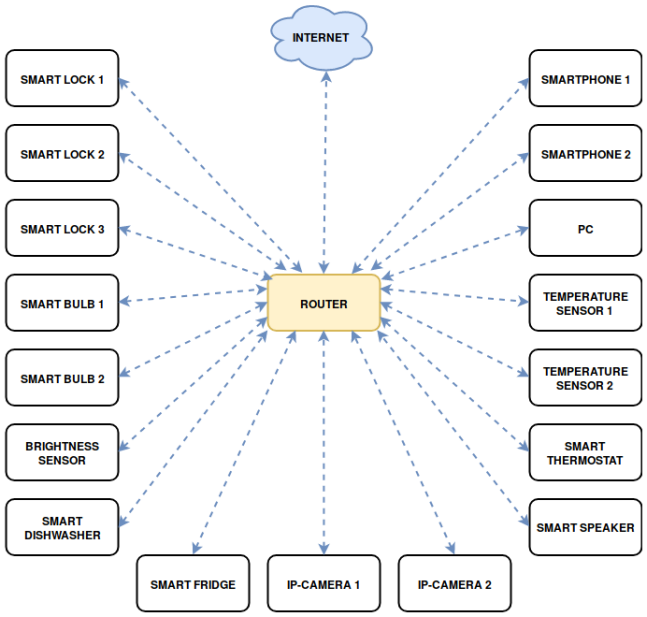


Figure 10 Schema of the scenario with star architecture

We have decided to develop two different scenarios composed of the same devices but adopting two different topologies. The first scenario, very similar to most current home networks is built using a star architecture, where the router is used as a communication interface both for the communication between all devices in the network and for communication from the outside to the inside of the network and vice-versa . In the second scenario, we have tried to apply traditional network security techniques to the home network and we have compartmentalized the home network so that it is divided into sub-networks, using hub devices, with the goal of limiting paths that an attacker could use.

The first scenario, reported in Figure 10, is composed of seventeen devices that communicate with each other and with the outside world using the router. Most of the devices are placed inside the physical building that is the house, hence there is no possibility for the attacker to physically tamper with any of the devices but the three smart-lock that are placed outside of the house. The smart-lock in the scenario are placed in this way: the device, named *SMART-LOCK 1*, permits the access to the area where are placed the other two smart-locks. The other two smart-locks permit the access to the internal areas of the house; hence for attackers to enter inside of the house both the *SMART-LOCK 1* and one of the other two smart-locks have to be compromised.

In the second scenario, represented in Figure 11, we compartmentalized the network; our aim here is to reduce the possible paths that an attacker could use by limiting the communication patterns among the devices; hence, we decided to divide the network into areas of use, creating the following sub-networks:

- *IP-camera Hub*: where the three IP-cameras are connected;

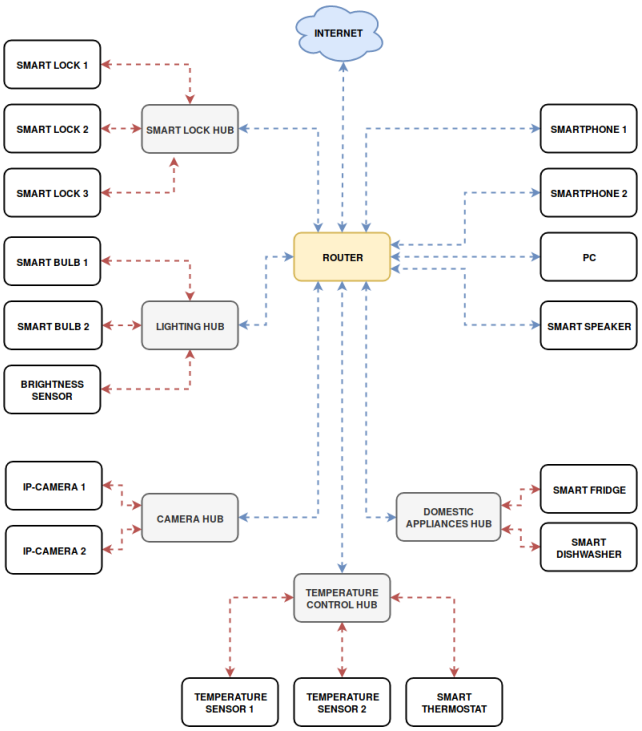


Figure 11 Schema of the scenario with hub architecture

- *Lock Hub*: where the three smart-lock are connected;
- *Lightning Hub*: where the brightness sensor and two smart bulbs are connected;
- *Smart Appliance Hub*: where the smart-fridge and the smart-dishwasher are connected;
- *Thermostat Hub*: where the thermostat and the two temperature sensors are connected;
- *Main Sub-network*: managed by the main router, where the PC, the two smart-phones (when present in the house) and the smart-speaker are connected.

In order to limit the paths that are not used by any legitimate application, we have decided to define a list of legitimate paths, while all the other paths are considered illegal. All the paths that are contained within a sub-network are legal, while the only legitimate paths that link different sub-networks are the following:

- *Smart-phones to Everything:* because smart-phones can control all the devices in the network;
- *PC to Everything:* as smart-phones, the PC can control all the devices in the network;
- *Smart speaker to Everything:* as previously described for the smart-phone and the PC, otherwise the feature of the smart-speaker would be limited.
- *Smart dishwasher to Smart-bulbs:* this path is legal, because the dishwasher can consult the state of the bulbs to decide if to start the washing or not.

The last path is just a possible example of an interaction between different categories of devices that can enhance the service provided. Of course, more can be defined and added to the scenario. In the development of the attack graphs, not all the permutations of the vulnerability are considered: in the graph there is only a path that represents all the others permutations, because the permutations do not add information to the graph. In order to build the graph we have chosen vulnerabilities from the database that were relevant to the devices in the scenario, every device has one or more vulnerabilities that an attacker could leverage to intrude the network. As described in the previous section, to each of these vulnerabilities a value is assigned: this value is relative to the potential of a specific vulnerability. This value is obtained using the CVSS Calculator, that on the base of some characteristics of the vulnerability gives in output three values: the overall score, the exploitability (i.e. the ease to use a specific vulnerability), and the damage that can be caused exploiting a specific vulnerability. In the graph, we have chosen to use the exploitability value as edge weight, because vulnerabilities are used as steps to access the target node, not as goals by themselves. Hence, the value of the damage related to a specific vulnerability is not important to our study. In the case of an edge that does not represents the exploitation of a vulnerability but just a simple action to be performed by the attacker we assigned the maximum value. The exploitability assumes a value from 0.1 to 3.9 where 0.1 indicates the lowest ease to use a specific vulnerability while 3.9 represent the highest level ease. The value of the exploitability depends on these factors:

- *Attack Vector:* thanks to this field, it is possible to set from where the attack start (e.g: Network, Adjacent Network, Local or Physical);
- *Attack Complexity:* with this attribute, it is possible to specify whether the attack is easy or difficult for the attacker;
- *Privileges Required:* it is possible to set what level of privileges are required to perform the attack;
- *User Interaction:* this field specifies whether the intervention of a third person is necessary for the success of the attack or not (i.e: the victim).

For all the vulnerabilities that are in in the database NVD, it is not necessary to execute the computation, because the database provides the value of the exploitability for every entry, represented by a unique code in the database. For the vulnerabilities that are not in the database, we have performed the computation inserting the attack characteristics in the CVSS calculator. In Table 1 we reported all the values gathered from the database; besides, for the computed vulnerabilities are also reported the vectors that represent the choices that we have made.

Table 1 Table with the value of the score relatives to every vulnerability in the scenario

Vulnerability Code	Exploitability CVSS Score	Normalized Score
CVE-2017-8221	3.9	1
CVE-2017-8224	3.9	1
CVE-2018-5383	1.6	0.39
CVE-2017-2872	1.2	0.29
CVE-2018-9995	3.9	1
CVE-2013-6952	3.9 AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N	1
CVE-2018-10561	3.9	1
CVE-2018-9158	3.9	1
CVE-2018-11629	3.9	1
CVE-2016-0019	2.2	0.55
CVE-2016-0117	1.8	0.45
CVE-2017-0783	2.8	0.71
EDB-45623	3.9 AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	1
EDB-44781	2.2 AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:L/A:L	0.55
POS-EXPLOIT	2.8 AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:N/A:N	0.71

In Figures 12 and 13 are represented the two attack graphs related to the two developed scenarios. It is possible to notice that in the graph of the partitioned scenario there is a lower number of edges because the information flow is limited by the hubs. In the simulation phase we studied if this second solution is more secure than the first one and how more secure it is, relating this value to the higher cost due to the network devices.

For a further study of the attack paths in the IoT Scenario, we have also tried to add other vulnerabilities to observe the change in the attack graph. The first of the vulnerabilities that we have added is an exploit that allows an attacker taking control of a smart-phone. This vulnerability, called HACK-EXPLOIT, has a value of 1.6 (normalized: 0.39); with the unaware cooperation of the victim, HACK-EXPLOIT allows an attacker to take control of the smart-phone and subsequently of the devices in the IoT network, trough the installation of a malware. Both in the star scenario and in the scenario with compartmentalizing hubs, the attack graph undergoes an important change; in fact, there is a new path, highlighted in red in the bottom of the graph, from the attacker to the goal state, shorter than the original paths. For the second vulnerability, we decided to use smart-locks with a USB-port, that the attacker could use to hack the device. This vulnerability is called PHYS-EXPLOIT and it has a value of 0.5 (normalized to 0.11), the value is small because this type of vulnerability requires physical access to the device. The addition of this new vulnerability does not introduce a substantial change in the attack graph; in fact, the new attack graph differs from the original one only for the presence of a new path that can be used by the attacker to obtain the control of the smart-lock, while the part of the graph that relates to the acquisition of the control of the IP-camera and the gathering of information about the house tenants are not modified. The last vulnerability we introduced is similar to the second one as it requires using the USB-port to install a crafted firmware to take control of a device, namely one of the IP-cameras. We have hypothesized the presence of an IP-camera outside the house, to make such an attack possible. The name of this vulnerability is CAMERA-EXPLOIT and its value is the same as the second one we described because, again, it requires physical access to the device. This vulnerability has a different effect on the attack graph; in the first scenario, where there are no limitations in the information flow, there is now another path from the attacker to the goal state. On the contrary, in the compartmentalized scenario, there are no significant changes to the attack graph.

Figures 14 and 15 illustrate the attack graphs with the added vulnerabilities; as we just described the three new vulnerabilities have different effects on the two attack graphs. The first vulnerability implies a significant change in both of the scenarios, adding a new path, highlighted in red in the graphs, shorter than the original one. The second vulnerability has not a clear effect onto the graphs, because the addition of a new path to achieve smart-lock access does not affect the overall graph. Instead, the last vulnerability has two different behaviors due to the information flow limitation. Indeed, in figure 14 there is a new path in the top section of the graph, while in figure 15 there is no change because the new path is prevented by the rules imposed by the network compartmentalization.

6 Experimental Results

To perform the analysis we followed this procedure: a random graph was created starting from Figure 7, the weight of each arc was calculated and inserted after which the network was saved (see Figure 16).

Then, the Topological analysis, the Extreme Pathway analysis, and the Flux Balance analysis were performed. After saving the obtained results from the analysis of the original network, nodes 4 and then the 3 were deleted

Table 2 Table with analysis results.

Network Analyzed	Hub Nodes	Extreme Pathways Number	Nutrient Uptake
Original Network	2 and 3	7	0.11
Network without node 4	2 and 3	6	0.11
Network without node 3	2 and 7	4	-

(attacks) in order to repeat the analysis and compare obtained results of the three different networks. From these analyses it is possible to identify the sequences of exploits that optimize the damage; furthermore, it is also possible to identify the minimum number of arcs (i.e., the minimum number of vulnerabilities of the network) which, if eliminated, would make the network partially sure.

The analysis of the original network provides the results shown in Table 2. A first result is the fact that the connectivity is high for node 2, in other words this fact makes multiple attack paths possible in the network. A second result is the identification of nodes that have the role of "connector hubs" [32]. In the analyzed network, these "connector hubs" are nodes 2 and 3. In the analyzed network there are 7 extreme pathways and a best path with a nutrient uptake of 0.11. Deleting node 4, the exploit that allows host 1 administrator privileges, does not significantly change the results of the analysis. On the contrary, by deleting node 3, which is a hub node, some differences in the results of the analysis can be observed. Nodes 2 and 7 become the new hub nodes, both with the role of "connector hub". Besides, the number of extreme pathways change from 4 to "no best path is found".

With more details, the weight of each arc of the network represents the risk that an exploit can cause a certain damage equal to the CVSS score of the exploit itself. The best path represents the path, which is the sequence of exploits, which maximizes the risk. This represents the sequence of exploits that maximizes both the damage suffered and the likelihood of suffering such damage. Examining the original network we observe that nodes 2 and 3 are hub nodes. The number of extreme pathways found is equal to 7 and the nutrient uptake of the best path is equal to 0.11. The best path consists of the following sequence of exploits:

$$IIS_BOF(0.0) -> Netbios_ssn_null_session(0.2) ->$$
$$Ftp_rhost(2.1) ->$$
$$-> Rsh_login(1.1) -> Squid_portscan(1.3) ->$$
$$LICQ_remote_to_user(1.3) ->$$
$$-> LOCAL_SETUID_BOF(3.3)$$

The route corresponds to taking ownership of host 0 through a buffer overflow, obtaining anonymous access to host 2 by leveraging a NetBIOS vulnerability, gaining host 1 confidence, and logging in it through the reverse shell, using Squid to make the scanner using open ports in machine 3 obtaining remote control of the machine and performing a local buffer overflow by obtaining host 3 Administrator privileges. By deleting node 3, no best path is found, this result can be interpreted as a state in which a potential hacker cannot find a sequence of exploits that maximizes the risk. By deleting node 4, the number of extreme pathways varies slightly while the best path does not vary. For a computational point of view, the proposed approach and toolset allow obtaining results in real time. For the network we just analyzed, the run time using a PC with an i7 CPU and 4GB of RAM running Windows 8.1 is just 5 seconds. As our aim is to make the network design and analysis tool as interactive as possible, we consider the time required to perform the analysis when the net is modified inside the tool itself; hence, we do not take into account all the startup overheads such as the time required to load libraries. Breaking down the total amount of time we observe that 0.16 seconds are employed by the topological analysis, 0.67 seconds are employed by the ExPa algorithm, while 2 seconds are employed by the FBA (measured by Matlab functions). We can compare our results with some data regarding algorithm performance provided in papers present in the literature. The authors in [14] show a computational cost algorithm of $M * N^3$ where M is the number of exploits and N is the number of hosts present. The computational cost of the extreme pathway is equal to N^3 with N number of graph nodes. Because the presence of each node is not dependent on the exploits in the hosts number but on states number in which the network can be found, the algorithm computational cost is not easy to calculate. An estimate can be obtained by assuming the best case in which each exploit allows to reach only one state, in this case, at most, the result is k*m knots, with m the number of exploits and K integer. The application of this algorithm is useful if the objective is used to get all paths that an attacker can follow to get own goal.

7 Conclusions

In this paper we have proposed a bio-inspired approach, based on algorithms used to study the metabolic network behavioral, to tackle the problem of attack graphs analysis. Mapping of some typical security metrics onto the biochemical quantities used in metabolic networks has been proposed, an automated tool has been implemented and its use has been described by means some IoT scenarios as use cases. The mapping and the usage of the tool allowed us to obtain values related the greatest risk and the minimum number of changes needed to morph the network into a new version that could be considered at least partially safe. With more details, the application of this bio-inspired approach allowed us to identify all paths that an attacker can follow to get own goal and to identify the most efficient way to thwart the attacker malicious will. The computational complexity of the used algorithms allowed observing the results of network modification in real-time, however, in more complex cases, additional optimization might be needed to keep this real-time behavior. Such optimization, together with the study of additional types of analysis will be the object of future work. Obviously, the security-to-metabolism mapping proposed in this paper is not the only one possible; in future work we plan to analyze additional mappings to evaluate additional aspects of IoT systems security and additional critical issues such as energy awareness.

References

[1] N. Gobbo, A. Merlo, and M. Migliardi. A denial of service attack to gsm networks via attach procedure. In A. Cuzzocrea, C. Kittl, D.E. Simos, E. Weippl, and L. Xu, editors, *Security Engineering and Intelligence Informatics*, pages 361–376. Springer Berlin Heidelberg, 2013.

[2] S. Al-Haj Baddar, A. Merlo, and M. Migliardi. Anomaly detection in computer networks: A state-of-the-art review. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 5(4):29–64, December 2014.

[3] Alfredo De Santis Rosario Russo Arcangelo Castiglione, Paolo DArco. Secure group communication schemes for dynamic heterogeneous distributed computing. *Future Generation Computer Systems*, 74:313–324, 2017.

[4] M. Pasha, G. Qaiser, and U. Pasha. A critical analysis of software risk management techniques in large scale systems. *IEEE Access*, 6:12412–12424, 2018.

[5] A. Armando, G. Pellegrino, R. Carbone, A. Merlo, and D. Balzarotti. From model-checking to automated testing of security protocols: Bridging the gap. 7305 LNCS:3–18.

[6] M. Ficco. Security event correlation approach for cloud computing. *International Journal of High Performance Computing and Networking*, 7(3):173 – 185, 2013.

[7] Aniello Castiglione Francesco Palmieri Ugo Fiore Arcangelo Castiglione, Alfredo De Santis. An energy-aware framework for reliable and secure end-to-end ubiquitous data communications. *Proceedings of the Seventh International World Wide Web Conference*, pages 107–117, 2013.

[8] Ugo Fiore Aniello Castiglione Alfredo De Santis Arcangelo Castiglione, FrancescoPalmieri. Modeling energy-efficient secure communications in multi-mode wireless mobile devices. *Journal of Computer and System Sciences*, 81(8):1464–1478, 2015.

[9] M. Migliardi and A. Merlo. Modeling the energy consumption of distributed ids: A step towards green security. pages 1452–1457, 2011.

[10] I. Chokshi, S.K. Das, N. Ghosh, S.K. Ghosh, A.K. Kaushik, and M. Sarkar. Netsecuritas: An integrated attack graph-based security assessment tool for enterprise network. *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, 2015.

[11] J.J. Treninen and R. Thurimella. Application of the pagerank algorithm to alarm graphs. *Computer Networks and ISDN Systems*, 4861:480 – 494, 2007. Information and Communications Security, ICICS 2007.

[12] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[13] Sumit Jalan, Pankaj Kumar, and Suvrojit Das. Formalization of digital forensic theory by using buchi automaton. *2015 Third International Conference on Image Information Processing*, pages 102–108, 2015.

[14] N. Ghosh and S. K. Ghosh. An intelligent technique for generating minimal attack graph. *First Workshop on Intelligent Security (Security and Artificial Intelligence) SecArt*, 2009.

[15] S. Noel, L. Wang, A. Singhal, and S.A. Jajodia. Measuring security risk of networks using attack graphs. *Proceedings of the International Journal of Next-Generation Computing*, 1(1), 2010.

[16] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. *Lecture Notes in Computer Science: Data and Applications Security XXII*, 5094, 2008.

[17] S. Vitabile, V. Conti, B. Lanza, D. Cusumano, and F. Sorbello. Metabolic networks robustness: Theory, simulations and results. *Journal of Interconnection Networks*, 12(3):221–240, 2011.

[18] V. Conti, S.S. Ruffo, S. Vitabile, and L. Barolli. Biam: a new bio-inspired analysis methodology for digital ecosystems based on a scale-free architecture. *Soft Computing*, pages 1–18, 2017.

[19] G. Vitello, A. Alongi, V. Conti, and S. Vitabile. A bio-inspired cognitive agent for autonomous urban vehicles routing optimization. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1):5–15, 2017.

[20] D.J. Kramer. Attack-defence graph - on the formalisation of security-critical system. *Master's Thesis - Saarland University, Germania*, 2015.

[21] M. Audinot, S. Pinchinat, and B. Kordy. Guided design of attack trees: A system-based approach. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 61–75, 2018. Proceedings of IEEE 31st Computer Security Foundations Symposium.

[22] G. Chen and Y. Huang. Research on an attack knowledge based enhanced attack tree model. *2011 International Conference on Computer Science and Service System (CSSS)*, 2011. Proceedings of International Conference on Computer Science and Service System (CSSS).

[23] A. Merlo, M. Migliardi, and P. Fontanelli. On energy-based profiling of malware in android. pages 535–542, 2014.

[24] R. Khoussainov and A. Patel. Lan security: problems and solutions for ethernet networks. *Computer Standards & Interfaces*, 22(3):191 – 202, 2000.

[25] P. Feng. Wireless lan security issues and solutions. In *2012 IEEE Symposium on Robotics and Applications (ISRA)*, pages 921–924, June 2012.

[26] V. Lacroix, L. Cottret, P. Thbault, and M.F. Sagot. An introduction to metabolic networks and their structural analysis. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 5(4):594–617, 2008.

[27] C. Schilling, D. Letscher, and B. Palsson. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *Journal of theoretical biology*, 203(3):229–248, 2000.

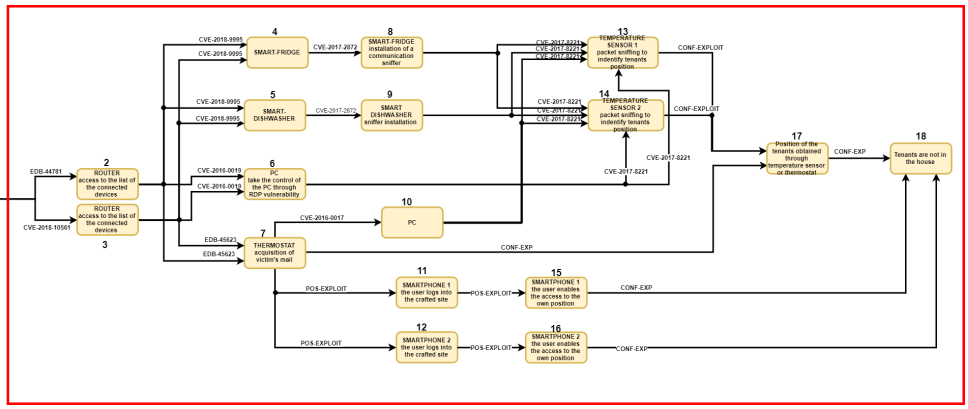
[28] K.J. Kauffman, P. Prakash, and J.S. Edwards. Advances in fux balance analysis. *Current Opinion in Biotechnology*, 14(5):491–496, 2003.

[29] Vincenzo Conti, Simone Sante Ruffo, Alessio Merlo, Mauro Migliardi, and Salvatore Vitabile. A bio-inspired approach to attack graphs analysis. In Arcangelo Castiglione, Florin Pop, Massimo Ficco, and Francesco Palmieri, editors, *Cyberspace Safety and Security*, pages 63–76, Cham, 2018. Springer International Publishing.

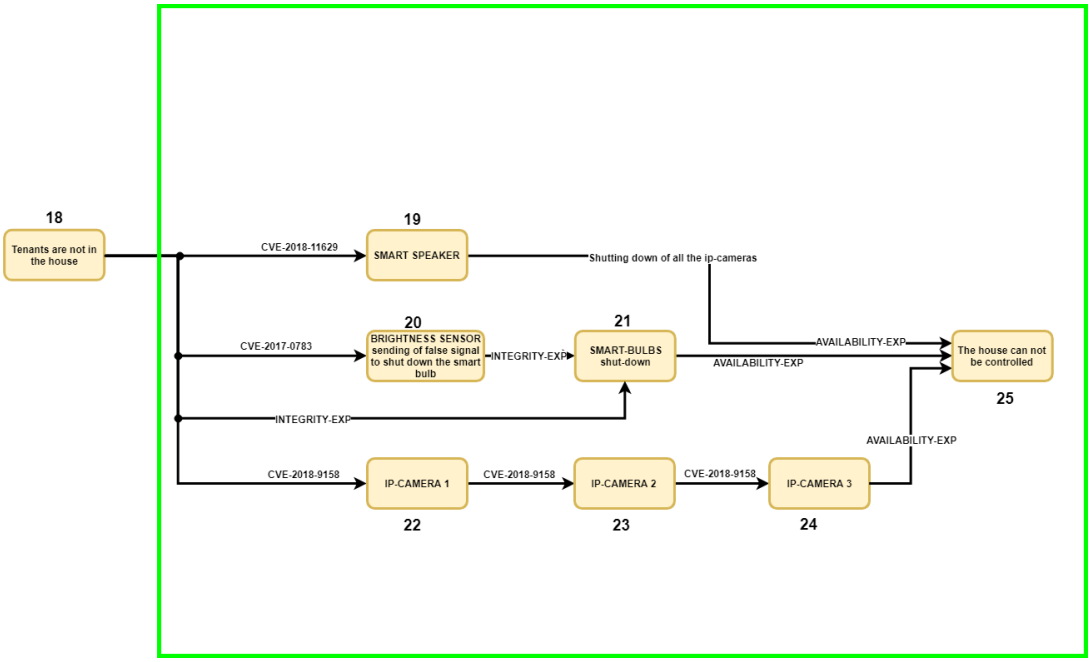
[30] NIST. Common vulnerability scoring system version 3.0 calculator, 2017.

[31] Center US-CERT. Security Operations NVD-2017, 2017.

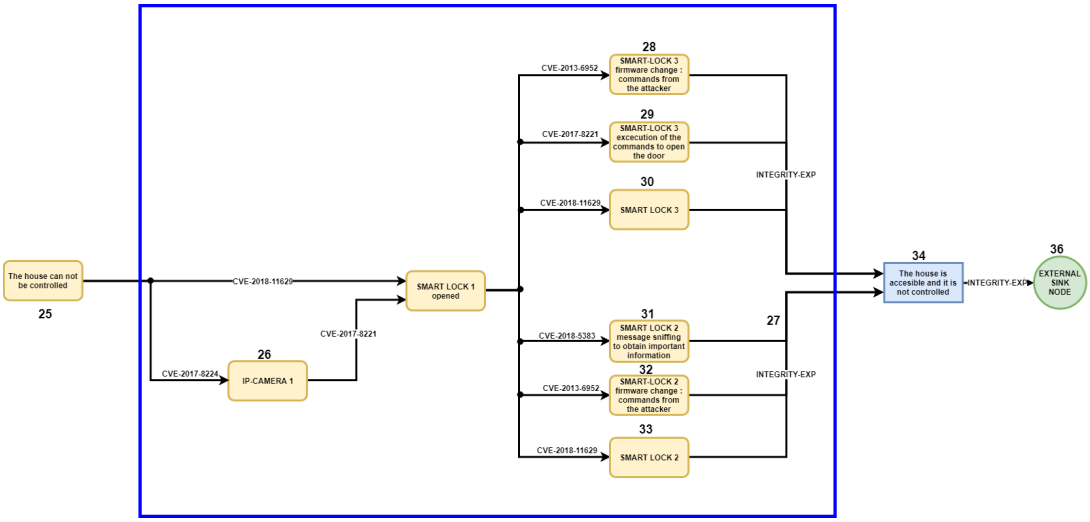
[32] R. Guimer and L.A.N. Amaral. Functional cartography of complex metabolic network. *Nature*, 433:895–900, 2005.



(a) First phase

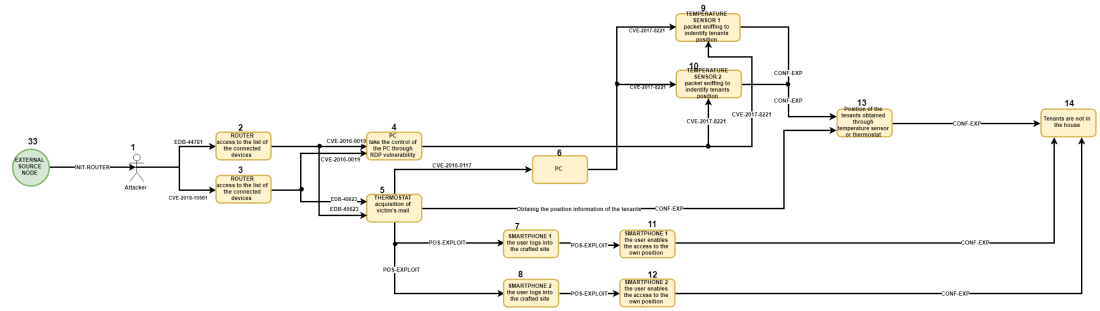


(b) Second phase

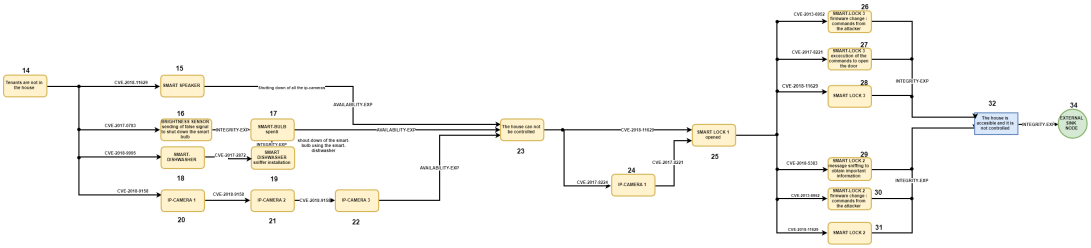


(c) Third phase

Figure 12 Attack Graph for the star architecture scenario

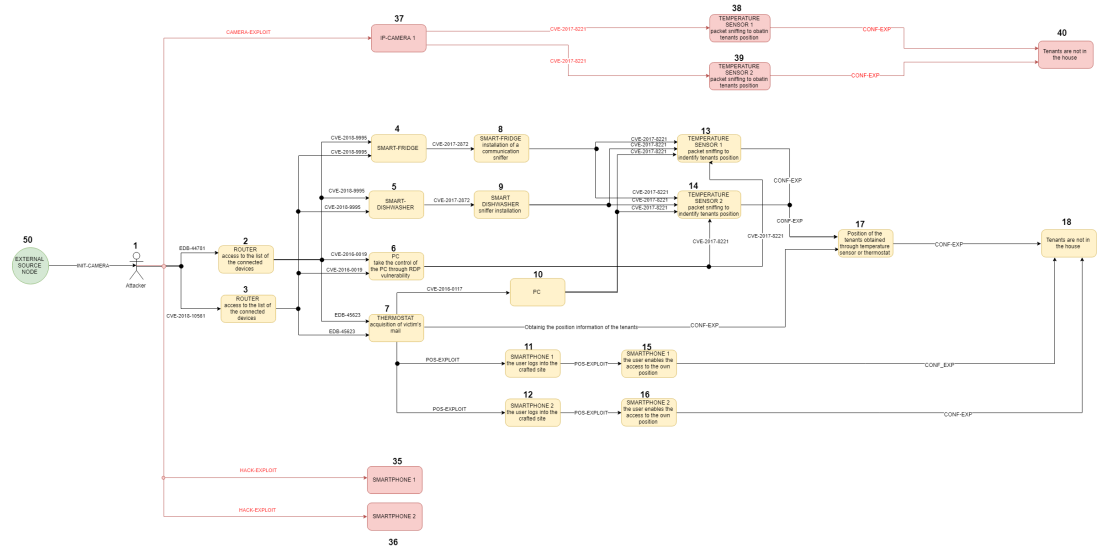


(a) First phase

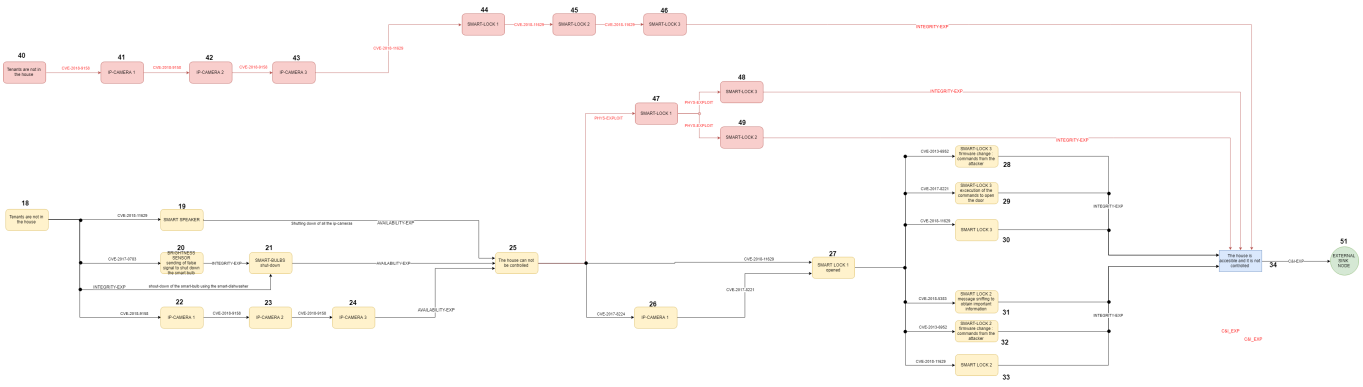


(b) Second phase

Figure 13 Attack Graph for the partitioned scenario

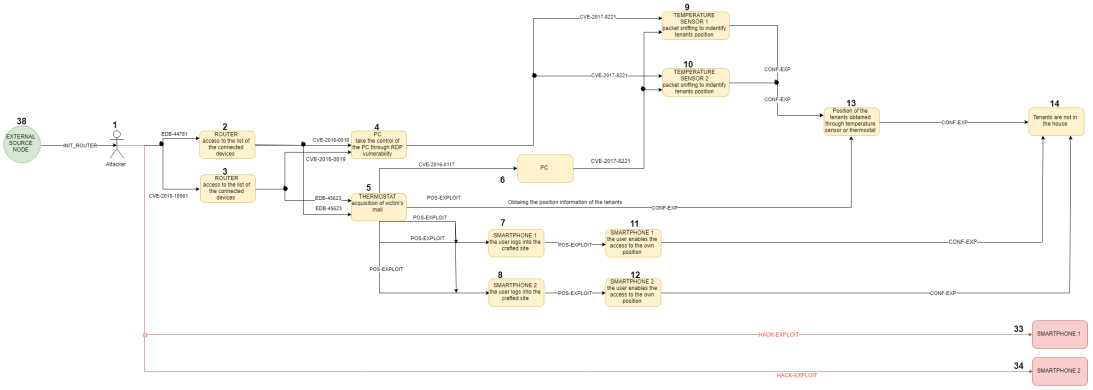


(a) First phase

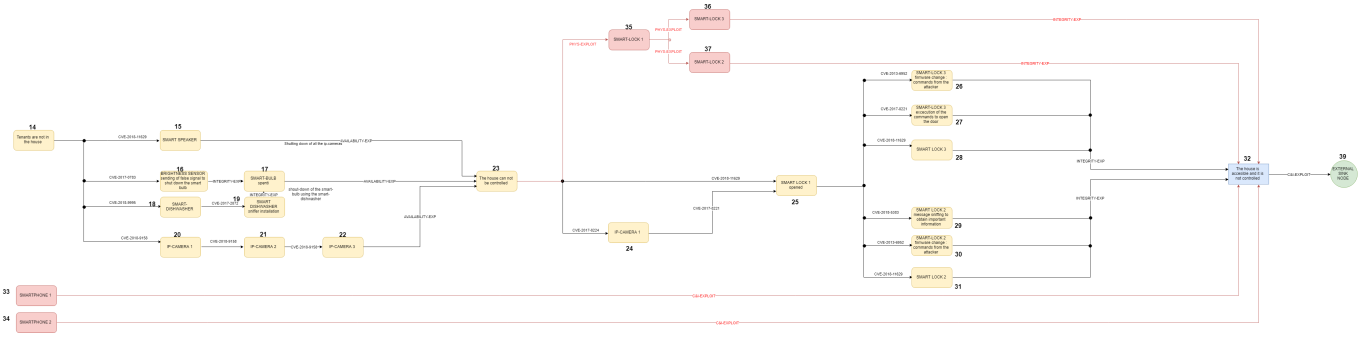


(b) Second phase

Figure 14 Graph attack of the star architecture scenario with the added vulnerabilities



(a) First phase



(b) Second phase

Figure 15 Graph attack of the partitioned scenario with the added vulnerabilities

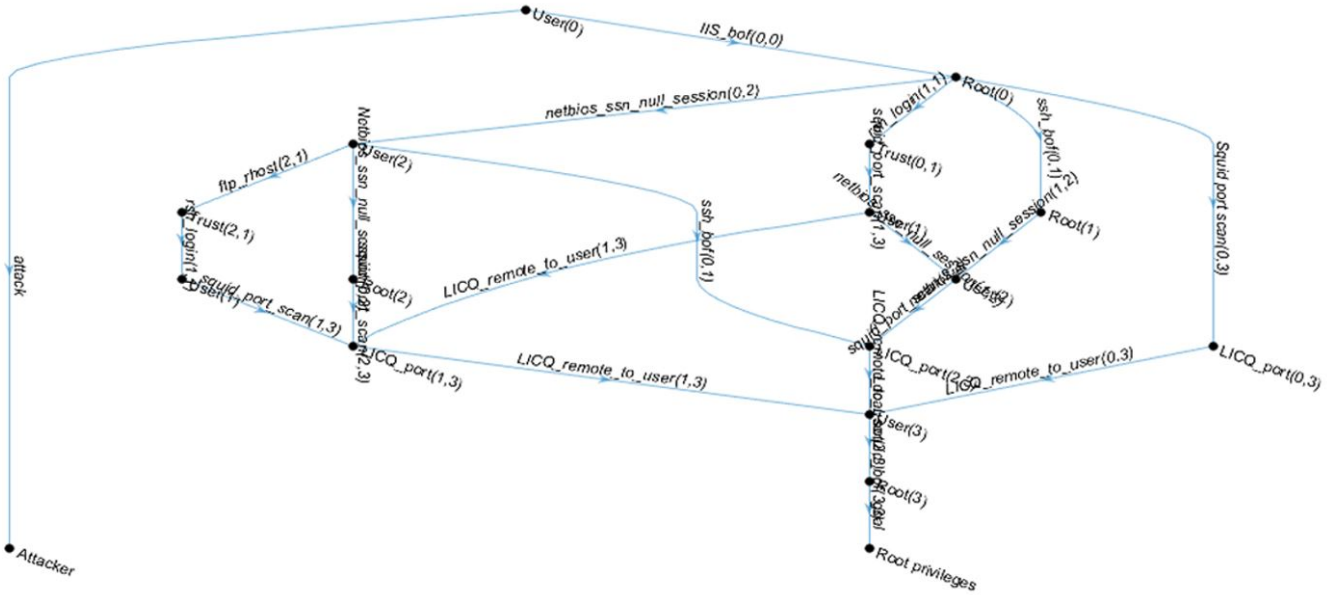


Figure 16 The analyzed network in this work