TÉCNICO LISBOA

Calculating Business Impact Assessment of Cyber-Threats

Diogo Martins Alves

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Dr. António Manuel Raminhos Cordeiro Grilo
Eng. Filipe Miguel Marcos Apolinário

Examination Committee

Chairperson: Prof. Dra. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Dr. António Manuel Raminhos Cordeiro Grilo
Member of the Committee: Prof. Dr. Carlos Nuno da Cruz Ribeiro

November 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to express my gratitude to all of those who have helped me in the development of this work.

Firstly, to Instituto Superior Técnico, not only for making this dissertation possible, but also for the education of excellence during these last five years.

I would like to thank my supervisors Prof. António Grilo and Eng. Filipe Apolinário for their guidance, support and motivation during the development of this thesis.

Also, I want to to thank the whole team at INOV, that helped me with their feedback and expertise, and made me feel integrated in the team.

Lastly, I would like to thank my family and friends for their unconditional support, with a special acknowledgement to my partner, Rita, for her encouragement and kindness.

Resumo

As organizações têm-se tornado progressivamente mais dependentes de tecnologias de informação e comunicação para suportar as suas operações quotidianas, o que inclui o armazenamento e acesso a informação crítica. Infelizmente, esta dependência em sistemas TIC deixa as organizações vulneráveis a ciberataques, que podem causar danos graves aos seus processos-negócio. Ao estimar o impacto causado por um determinado ciberataque numa dada organização, é possível priorizar as ações de mitigação e prevenção a serem consideradas no processo de gestão de risco. Para além disso, uma metodologia capaz de estimar o impacto pode também ser útil na previsão de *Falhas em Cascata* que resultam das interdependências entre diferentes organizações.

Como resultado, é proposta a metodologia *Business Impact Calculator* (BusICalc). O BusICalc foi projetado para oferecer um método capaz de quantificar o impacto que uma ameaça causaria nos processos-negócio de uma organização. Um protótipo do BusICalc foi desenvolvido para a avaliação e integrado com o sistema de análise de risco, BIA (*Business Impact Assessment*). A metodologia proposta foi avaliada usando um *dataset* correspondente a uma *Infraestrutura Crítica*, e as experiências realizadas mostram a escalabilidade do BusICalc e a sua eficácia em gerar valores razoáveis para o impacto de ciber-ameaças.

Palavras-chave: Ciberataque, Propagação de impacto, Modelação de processos-negócio, Efeitos em cascata, Quantificação de impacto, Segurança

Abstract

Organizations are becoming increasingly more reliant on information and communication technology to support their day-to-day operations, which includes the storage and access of critical information. Unfortunately, this dependency on ICT systems leaves organizations vulnerable to cyber-attacks, which can cause serious damage to their business-processes. By estimating the impact caused by a given cyber-attack in a particular organization, it is possible to prioritize the mitigation actions and preventative measures to be considered in the risk management procedure. Moreover, a methodology capable of estimating impact can also be useful in predicting the *Cascading Failures* that result from the interdependencies between different organizations.

As a result, the *Business Impact Calculator* (BusICalc) methodology is proposed. BusICalc was designed to offer a method capable of quantifying the impact that a cyber-threat would cause, once exploited, to the organization's business-processes. A proof-of-concept of BusICalc was developed for evaluation purposes and integrated with the risk analysis system, BIA (*Business Impact Assessment*). The proposed methodology was evaluated using a dataset corresponding to a *Critical Infrastructure*, and the conducted experiments show that BusICalc is scalable and effective in yielding reasonable values for the impact of cyber-threats.

Keywords: Cyber-Attack, Impact Propagation, Business-Process Modelling, Cascading Effects, Impact Quantification, Security

Contents

List of Tables

List of Figures

Listings

List of Acronyms

AP Access Point

ATT&CK Adversarial Tactics, Techniques & Common Knowledge

BIA Business Impact Assessment

BN Bayesian Network

BP Business-Process

BusICalc Business Impact Calculator

CB Circuit Breaker

CI Critical Infrastructure

CISA Cybersecurity and Infrastructure Security Agency

CoDeSys Controller Development System

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DAG Directed and Acyclic Graph

DoS Denial of Service

DREAD Damage, Reproducibility, Exploitability, Affected users, Discoverability

EMP Electromagnetic Pulse

EPIC Electric Power Intelligent Control

FF Fraction of Functional Entities

FRAM Function Resonance Analysis Method

ICS Industrial Control System

ICT Information and Communication Technology

IED Intelligent Electronic Devices

IF Impact Factor

INOV Instituto de Engenharia de Sistemas e Computadores Inovação

IT Information Technology

MulVAL Multihost, Multistage Vulnerability Analysis

NE Network Efficiency

NIST National Institute of Standards and Technology

NVD National Vulnerability Database

OC Operational Capacity

OT Operational Technology

OWF Offshore Wind Farms

PLC Programmable Logic Controller

PV Photovoltaic

RAG Risk Assessment Graph

SCADA Supervisory Control and Data Acquisition

SSR Safety, Security and Resilience

SW Switch

TCP Transmission Control Protocol

UDP User Datagram Protocol

VASM Vulnerability, Asset, Service, Mission

WS Workstation

Chapter 1

Introduction

1.1 Motivation

As ICT (Information and Communication Technology) systems become more common in the control and monitoring of *Critical Infrastructures* (such as energy and water distribution, transportation, communications), the risk of cyber-attacks capable of compromising the operations of such infrastructures increases [1]. Moreover, considering that there are *interdependencies* between different infrastructures (i.e., relationships through which the state of one infrastructure is influenced by or correlated to the state of other infrastructure [2]), the compromise of the operations of one *Critical Infrastructure* can, in turn, cause failures in other infrastructures that are dependent on the first, in a process known as *Cascading Failures*.

The increased demand for the services (e.g., basic resources (power, gas, water), communications (telephone, Internet), transportation (land, air, sea) [3]) provided by *Critical Infrastructures* (CIs) has forced them to become more automated, by increasing their OT (Operational Technology) and IT (Information Technology) infrastructure with intelligent devices, such as Industrial Control Systems (ICS), capable of supervising the CI work and making informed decisions to maximize the business profit. Often these intelligent devices need to be connected to the Internet. This makes *Critical Infrastructures* a tempting target to cyber-attackers, due to the high reliance of modern society on the services provided by these infrastructures [1].

Examples of such attacks include the BlackEnergy, and Industroyer [4] malwares. The BlackEnergy and Industroyer were both responsible for cyber-attacks to the Ukrainian power grid, the first in December of 2015 and the second in December of 2016. In the first attack, BlackEnergy was able to exploit remote access software to cut off power to around 250 000 households for six hours. A year later, Industroyer managed to deprive Ukraine's capital, Kiev, of power for an hour by taking control of electricity substation switches and circuit breakers.

Due to the interdependencies between different *Critical Infrastructures*, even if an infrastructure is not the primary target of a cyber-attack, it can still be impacted if other infrastructures it depends on suffer failures. A real-life example is the Texas power outage of February 2021 [5]. In this case, the

outage was not due to a cyber-attack but instead to the lack of preparedness of electricity generation infrastructure to extremely cold temperatures, caused by severe winter storms. The shortage of power for several days led to cascading impacts in other infrastructures, mainly the water supply system and the food distribution system [6]. It is estimated that it was responsible for around 150 deaths [7] and around $200bn in economic losses [8]. A 2015 study [9] shows that the economic impact of a severe cyber-attack against the United States' power grid could total over $240bn, perhaps even reaching $1trn in the most extreme scenario.

By providing a methodology capable of estimating the impact caused by a given cyber-threat, it is possible to prioritize the threats that should be mitigated during risk assessment [10].

1.2 Objectives

The goal of this dissertation is the design and implementation of a methodology — BusICalc (*Business Impact Calculator)* — capable of quantifying the impact of the propagation of a cyber-threat across an organization. More specifically, the methodology should calculate the impact that a given cyber-threat can have, once exploited, on the critical business-processes of an organization. For instance, if the considered organization is a *Critical Infrastructure*, then its critical business-processes correspond to the correct delivery of the infrastructure's essential services to its customers, whose disruption would cause severe economic and/or reputational damage (e.g., in an electrical grid, the critical business-process of interest would be the reliable delivery of power to the grid's customers). Hence, this methodology would allow the identification of the most impactful threats to the organization, which in turn would contribute to prioritize the mitigation actions and preventative measures to be taken in the risk management procedure.

To do so, the BIA (*Business Impact Assessment*) methodology will be employed, in order to model the entities that comprise the organization and perform the simulations of the propagation of user-chosen cyber-threats. Hence, BusICalc's main objective will be providing a way to assign an impact metric to a BIA simulation.

1.3 Contributions

With the development of BusICalc, this work offers the following contributions:

- Calculation of an impact value (between 0 and 1) that evaluates the level of operationality loss suffered by the business-processes of an organization to a simulated attack. This value is computed by leveraging the structure of a given organization (i.e., the network connections between devices, the threats they are vulnerable to, the services they provide, and the interconnections between the activities that comprise the organization's business-processes) in order to simulate the propagation of a chosen entry-point threat.

- Development of a proof-of-concept that integrates the impact calculation methodology with an existing risk assessment system — BIA (*Business Impact Assessment*) ([11, 12]).

The evaluation process conducted to test the efficacy of BusICalc used a model of a small-scale electric smart-grid as the testbed, and BusICalc was shown successful in computing the impact of cyber-threats on the objectives of such system. Additionally, the application of the methodology to a critical business-process of a *Critical Infrastructure* shows that the compromise of these types of processes can lead to failures in other *Critical Infrastructures* (*Cascading Failures*). For example, a disruption in the delivery of power would affect the electric pumps, which would make the water distribution system inoperable.

For the evaluation of performance, simulations were performed on a testbed of varying size, and it was concluded that the execution time of BusICalc only increases linearly with the size of the network, making the solution scalable.

1.4 Structure of the Document

The remainder of this thesis is structured as follows: Chapter 2 presents a literature review on (1) methods for simulating *Cascading Failures*, as well as determining their impact, and (2) methodologies for impact assessment and propagation of cyber-threats; Chapter 3 explains the design process behind the development of the BusICalc methodology; Chapter 4 presents the implementation details of the development of this tool; Chapter 5 describes the experiments conducted to test BusICalc, as well as the scenario used for this evaluation; and Chapter 6 concludes the thesis, by summing up the main results of this work, and proposing ideas for future work.

Chapter 2

Literature Review

The goal of this work is to develop a methodology to calculate the impact of cyber-threats in business organizations. A possible area of application of this methodology is in the prediction of *Cascading Failures*. *Cascading Failures* are especially relevant in interdependent networks, i.e., networks of devices, where each device depends on communications to other devices to function correctly. This interdependency causes the network to be more fragile than a single isolated system since a failure in one device can propagate and cause failures in other devices that depend on it, which in turn can propagate to their dependent devices and spread even further [13] — *Cascading Failures*. The impact of a failure refers to a measure of its outcome. In the context of *Cascading Failures*, the impact of a failure must consider the whole cascading propagation that the failure may cause.

Cyber-threats represent one possible trigger for *Cascading Failures*. Not only can they spread easily across interconnected networks but they can also turn into physical threats in cyber-physical systems. In order to understand how a cyber-attack can create *Cascading Failures*, and how to quantify the extent of their impact, this chapter presents a literature review of current methods used for these purposes, divided into the following sections: Section 2.1 presents models used to simulate *Cascading Failures* in different types of systems (namely *Supply Chains* and *Critical Infrastructures*), and measure their impact; Section 2.2 analyses methods for determining the level of risk of software vulnerabilities, and assessing the impact propagation of cyber-threats that exploit such vulnerabilities, in information technology (IT) networks; finally, Section 2.3 presents a summary and discussion of all the methods mentioned in the two previous sections.

2.1 Cascading Failures

Cascading Failures can occur in different types of networks (e.g., power grids ([14, 15]), cyber-physical systems ([13]), transportation systems ([16]), *Supply Chains* ([17–19]), etc.). In the literature, *Cascading Failures* have been studied in primarily two domains: *Supply Chains* (Section 2.1.1) and *Critical Infrastructures* (Section 2.1.2).

A *Supply Chain* is a network of corporations (e.g., suppliers, manufacturers, distributors, and retail-

ers) that cooperate among each other by providing services, such as purchasing materials, processing materials into products, and delivering products to customers [20].

A *Critical Infrastructure* (CI) is a system considered so essential that its failure would have significant effects on public health, safety, or economic security. The Cybersecurity and Infrastructure Security Agency (CISA) [21] defines a collection of 16 sub-sectors that can be classified as *Critical Infrastructures*, which includes, among others, energy, water supply, transport, and communications.

2.1.1 Cascading Failures in Supply Chains

In the works that address *Cascading Failures* in *Supply Chains* ([17–19]), the used model for the *Supply Chain* network consists of a directed graph, in which each node represents a corporation and each edge represents a commercial relationship between two corporations, i.e., in edge ab, directed from a to b, b is the client of a and a is the supplier of b. Each edge, ab, has a weight (y_{ab}) that represents the strength of the commercial relationship between the two corporations. Each node is characterized by an upper-bound (B^{upper}) and a lower-bound (B^{lower}) for its load (L). In a *Supply Chain* network, the load often represents material flows (i.e., transport of materials, components, or products), but it can also represent capital flows (i.e., circulation of funds) or information flows (i.e., the transmission of collaborative data). The direction of the edges (from suppliers to clients) represents the direction of load (e.g., flow of materials). This does not mean, however, that a client cannot influence its suppliers. In fact, when there is a loss of load in a client, its suppliers will also suffer such a loss.

Whenever a node fails, either by external reasons (not represented in the graph) or due to *Cascading Failures* from another node failure, that node is eliminated from the graph, along with the edges connected to it. A failure of a node means that the corporation it represents can no longer stay in business, so it is removed from the model and its commercial relationships (as supplier and customer) cease to exist. As a consequence, the loads and edge weights of the remaining loads need to be updated, based on the topology of the network after the failure.

Cascading Failures caused by underload have been studied by **Wang and Zhang [17]** in the context of *Supply Chains*. Underload means that a node fails when its load (L) is smaller than its lower-bound capacity (B^{lower}). In practical terms, this means that either a decrease in the demand for the corporation's materials or a decrease in the supply of materials to the corporation has caused it to no longer remain in business.

In this model [17], the weight of a given edge that links a to b, y_{ab}, can be estimated by the Equation $y_{ab} = (z_a z_b)^{\lambda}$, where z_a and z_b are, respectively, the degree (number of neighbours) of nodes a and b; and λ is an adjustable parameter that defines the strength of the weights of the edges (in [17], $\lambda = 0.5$ was used). The weight of the edge represents the strength of the commercial relationship between two corporations. The initial load of each node a, L_a^0, is given by the Equation 2.1, where δ is an adjustable parameter, that defines the strength of the nodes' load, and $N(a)$ represents the set of nodes that are

neighbours (clients and suppliers) of node a.

$$L_a^0 = [z_a \sum_{i \epsilon N(a)} z_i]^\delta \tag{2.1}$$

Each node starts with a load equal to its initial load and the load does not change unless there is a failure in the network. The upper and lower bounds of each load are proportional to its initial load and are given, respectively, by the Equations $B_a^{upper} = \alpha L_a^0$ and $B_a^{lower} = \beta L_a^0$, with $\alpha \geq 1$ and $\beta \leq 1$.

Whenever a node fails, the load of its neighbours will be adjusted. For example, suppose that a supplier s has a client e. When supplier s fails, it will stop supplying materials to its client e, so the load of e is reduced by the quantity ΔL_e^- in Equation 2.2 (introduced in [17]), where $N^+(s)$ represents the set of clients of s and $N^-(s)$ represents the set of suppliers of s. ΔL_e^- represents the fraction of the load of s that was being directed to e. It is calculated by multiplying L_s by the fraction between the weight of edge se, y_{se}, and the sum of weights of all edges connected to s ($\sum_{d \epsilon N^-(s)} y_{ds} + \sum_{i \epsilon N^+(s)} y_{si}$).

$$\Delta L_e^- = L_s \frac{y_{se}}{\sum_{d \epsilon N^-(s)} y_{ds} + \sum_{i \epsilon N^+(s)} y_{si}} \tag{2.2}$$

Node e, which has seen its load now reduced, will try to increase it by resorting to one of its remaining suppliers. Assuming that there is a supplier c of e, this supplier has a redundant capacity, R_c, of $R_c = B_c^{upper} - L_c$, i.e., the redundant capacity corresponds to the capacity that is available to a node but is not in use. In case there is more than one supplier of e, node e will choose c with probability $P_c = \frac{R_c}{\sum_{i \epsilon N^-(e)} R_i}$. In case node c is indeed chosen, the load of e will be increased in the quantity $\Delta L_{ce}^+ = min(\Delta L_e^-, R_c)$, and the weight of the commercial relationship between c and e will also increase: $y_{ce} = y_{ce} \frac{L_c + \Delta L_{ce}^+}{L_c}$. At this time, it is necessary to update the value of the decrease of load in node e: $\Delta L_e^- := \Delta L_e^- - \Delta L_{ce}^+$ and update the weights of the commercial relationships between e and its clients — for each client g, $y_{eg} := y_{eg} \frac{L_e - \Delta L_e^-}{L_e}$. Node e will repeat this process until its load L_e is greater or equal to its lower-bound capacity (B_e^{lower}), or until it runs out of suppliers. In case it runs out of suppliers and its load is lower than its lower-bound capacity, then there is a failure in node e. Yang et al. [18] supplements this method by allowing nodes that are about to fail to establish new business relationships with suppliers/clients with which a business relationship did not previously exist and that still have redundant capacity.

This process shall be executed for every client of the node a that suffers the initial failure, as well as for its suppliers, in the opposite direction. It should be taken into account that whenever a successive failure occurs (for example a failure in node e), this process will have to be executed again, now starting at node e. The process only stops when there are no more failures or if there is a total failure of every node.

In order to evaluate the robustness of the network to failures, this method defines the Network Efficiency (NE), calculated by Equation 2.3, where ϕ_{ab} is the efficiency between nodes a and b, which is equal to the sum of edge weights (y) on the shortest path between a and b, and M is the total number of nodes in the network. This parameter can be used to measure the consequences of the *Cascading Failures*, since the greater the decrease in the value of NE (from before to after the failures), the more

severe are the *Cascading Failures*.

$$NE = \frac{\sum_{a,b,a\neq b} \phi_{ab}}{M(M-1)} \tag{2.3}$$

Zeng and Xiao [19] propose a similar method, given that it uses the same graph model as [17] to represent the network of corporations, but with some important distinctions — firstly, the initial load of each node is not based on its number of neighbours but instead on its betweenness. The betweenness ($C_B(i)$) of a node (i) measures the centrality of the node in the graph. More specifically, it measures the fraction of shortest paths between every pair of nodes in the graph that contain node i. So, if the betweenness of i is close to 1, it means that i is a very central node in the graph, i.e., almost every (shortest) path between any two nodes will contain the node i. On the other hand, if the betweenness of i is close to 0, it means that i is a peripheral node, i.e., almost none of the shortest paths in the graph contain i. The betweenness of a node i, $C_B(i)$, is given by Equation 2.4, where σ_{kl} is the total number of shortest paths between nodes k and l, $\sigma_{kl}(i)$ is the number of shortest paths between nodes k and l that go through i, and n is the total number of nodes in the graph.

$$C_B(i) = \frac{\sum_{k\neq l} \frac{\sigma_{kl}(i)}{\sigma_{kl}}}{n(n-1)} \tag{2.4}$$

The initial load (L_i^0) of each node i depends not only on its betweenness but also on the betweennesses of each of its neighbours, according to Equation 2.5, where μ is a tunable parameter that governs the strength of the initial load.

$$L_i^0 = (C_B(i))^\mu + (\sum_{m\epsilon N(i)} C_B(m))^\mu \tag{2.5}$$

In this model ([19]), the cause of node failure is overload rather than underload, i.e., what causes the failure is the excess of load rather than the lack of it, so whenever a node i fails, each of its neighbours, j, will see their load increase by the quantity ΔL_{ij}, given by Equation 2.6.

$$\Delta L_{ij} = \frac{L_j}{\sum_{k\epsilon N(i)} L_k} \times L_i \tag{2.6}$$

It is assumed that the nodes are able to withstand a certain amount of overload before they fail, i.e., nodes do not fail as soon as their load surpasses their upper bound ($L_i \geq B_i^{upper}$), but instead fail when the load surpasses the upper bound multiplied by the overloaded parameter, γ. In other words, the condition that causes node i to fail is $L_i \geq \gamma B_i^{upper}$. The overloaded parameter, γ, is a value greater than 1 that is a measure of the contingencies that the system employs in order to avoid *Cascading Failures*.

2.1.2 Cascading Failures in Critical Infrastructure Systems

Cascading Failures have also been studied in the literature in the context of *Critical Infrastructure* (CI) systems ([10, 13, 14, 16, 22–28]). Here, *Cascading Failures* can propagate from one infrastructure to the

next if the second has a dependency to the first. According to [29], two infrastructures are dependent if either (1) they share a component (e.g., the Water Distribution System and the Fire Emergency Services may share a water reservoir. In this case, the consumption of water in one infrastructure will affect the availability in the other); (2) one of the infrastructures provides an input to the other (e.g., the electricity generated by the Electric Power System is used as input by the Water Supply System) or (3) the two infrastructures compete for the flow of resources (e.g., if both the Railway Infrastructure and the Road Transport Infrastructure have a route from point X to point Y).

This section presents methods that simulate the spread of *Cascading Failures* in *Critical Infrastructure* systems and estimate their impact. The methods can be divided according to their scope: [10, 23, 24] focus on dependencies between different CIs and [16, 25–28] identify dependencies inside the CIs. Among the works that focus on dependencies inside CIs, it is possible to further distinguish between models that study physical dependencies ([16, 25, 26]), and models that study dependencies based on processes and functions of the CIs ([27, 28]).

Rehak et al. [23] proposes a method to assess and quantify the spread of *Cascading Failures* in CI systems. The spread of failures in a CI system results from the mutual links between individual sub-sectors. Each type of infrastructure (electricity, road transport, water supply, healthcare, etc.) is a sub-sector and there are links between sub-sectors when they have interdependencies. This spread is quite hard to predict since the topology of the CI system may be complex and contain subtle feedback loops, that can propagate disturbances in an unforeseeable way [30]. The spread, however, does depend on external and internal factors. External factors include the magnitude, type, and duration of the emergency, as well as the layout of the links in the system. As for Internal factors, the most important is the system's resilience. A system is said to be resilient if it can carry out its mission to some extent and recover within an acceptable amount of time when it suffers a major disturbance. A system needs to be resilient because it will eventually suffer disruption and when it happens, the lack of resilience will cause it to become inoperable [31].

The first step of the proposed framework [23] is to identify which are the sub-sectors belonging to the *Critical Infrastructure* in the case study scenario, as well as the dependencies between them by building a matrix of correlation between sub-sectors, from the identified links between them. This matrix will only have the values 1 or 0. A value of 1 in entry a_{ij} of the matrix means that the sub-sector S_i can cause the failure of sub-sector S_j and a value of 0 means that there is no real possibility of that happening.

Next, two coefficients are defined – activity coefficient (K_A), in Equation 2.7, and passivity coefficient (K_P), in Equation 2.8. The activity coefficient $(K_{A_{S_i}})$ measures the fraction of sub-sectors that can be affected if sub-sector S_i fails, and the passivity coefficient ($K_{P_{S_i}}$) measures the fraction of sub-sectors that can cause S_i to fail. The parameter n is the total number of sub-sectors.

$$K_{A_{S_i}} = \frac{\sum_{j=1}^{n} a_{ij}}{n-1} \tag{2.7}$$

$$K_{P_{S_i}} = \frac{\sum_{j=1}^{n} a_{ji}}{n-1} \tag{2.8}$$

Each of the sub-sectors needs to be evaluated in regards to a set of criteria, with values between 0 and 1, divided into three groups:

- The first group (C_T) relates to the intensity of the initiation threat, i.e., the severity posed by the threat (the severity is greater when values are closer to 1).
  - Activation Ability (C_{T_1}) — Amount of time available from the prediction of the threat to the actual impact (the shorter it is, the more severe is the threat).
  - Exposure (C_{T_2}) — Duration of the impact of the threat (the longer it is, the more severe).
  - Potential (C_{T_3}) — Intensity of the impact of the threat in this sub-sector and the next dependent sub-sectors.
- The second group (C_{EV}) measures the Resilience level of the sub-sector to the threat (the resilience is greater when values are closer to 0).
  - Availability (C_{EV_1}) — Probability that the sub-sector will be impacted.
  - Resistance (C_{EV_2}) — Level of security of the sub-sector (value closer to 0 means more secure).
  - Criticality (C_{EV_3}) — Level of importance of the sub-sector in the CI system.
  - Recoverability (C_{EV_4}) — Amount of time necessary to recover the damaged sub-sector.
- The third group (C_{EM}) refers to the level of security measures of the sub-sector (a value closer to 1 means more effective security measures).
  - Efficiency (C_{EM_1}) — Ability of the security measures to reduce the impact of the threat.
  - Feasibility (C_{EM_2}) — Availability of the security measures to increase security
  - Financial Demand (C_{EM_3}) — Availability of financial resources necessary to implement security measures.
  - Time Demand (C_{EM_4}) — Amount of time necessary to implement security measures (a value closer to 1 means less time).

The criteria C_T only needs to be defined for sub-sectors from which further *Cascading Failures* may spread and the criteria C_{EV} and C_{EM} for the sub-sectors to which failures can spread.

With these criteria, it is possible to define the Initiation Threat Intensity influencing the sub-sector S_i — T_i — and the Resilience Level of the dependent sub-sector S_i — E_i — which will be used to calculate the intensity of a cascading impact effect spreading from one sub-sector to another.

$$T_i = \frac{\sum_{k=1}^{n_T} C_{T_k;S_i}}{n_T} \tag{2.9}$$

$$E_i = 1 - \left(\frac{\sum_{k=1}^{n_{EV}} C_{EV_k;S_i}}{n_{EV}} - \frac{\sum_{k=1}^{n_{EV}} C_{EV_k;S_i}}{n_{EV}} \times \frac{\sum_{k=1}^{n_{EM}} C_{EM_k;S_i}}{n_{EM}}\right) \tag{2.10}$$

From this, it is now possible to define the Probability of the Cascading Impact Effect spreading from the sub-sector S_i to the sub-sector S_j (P_{ij}); the Intensity of the Cascading Impact Effect spreading from

the sub-sector S_i to the sub-sector S_j (I_{ij}); and the Risk of a Cascading Impact Effect spreading from the sub-sector S_i to the sub-sector S_j (R_{ij}).

$$P_{ij} = \frac{K_{A_{S_i}} + K_{P_{S_j}}}{2} \tag{2.11}$$

$$I_{ij} = T_i \times (1 - E_j) \tag{2.12}$$

$$R_{ij} = P_{ij} \times I_{ij} \tag{2.13}$$

The Risk of a Cascading Impact Effect spreading from the sub-sector S_i to the sub-sector S_j is the most important result from this method. It provides a measure of severity for a cascading impact that spreads through that link.

In this article [23], the example scenario is a Blackout in an industrialized region of the Czech Republic. The sub-sectors identified were: electricity, road transport, water supply, rail transport, integrated rescue system, and healthcare. The relationships between these sectors and the direction of impact cascade spread must also be identified. In Figure 2.1 these relationships are shown for this example. Here, the sub-sector electricity can propagate failures to the sub-sectors road transport and water supply; the sub-sector road transport, in turn, can propagate failures to the sub-sectors rail transport and integrated rescue system; and the water supply sub-sector can propagate failures to the healthcare sub-sector.

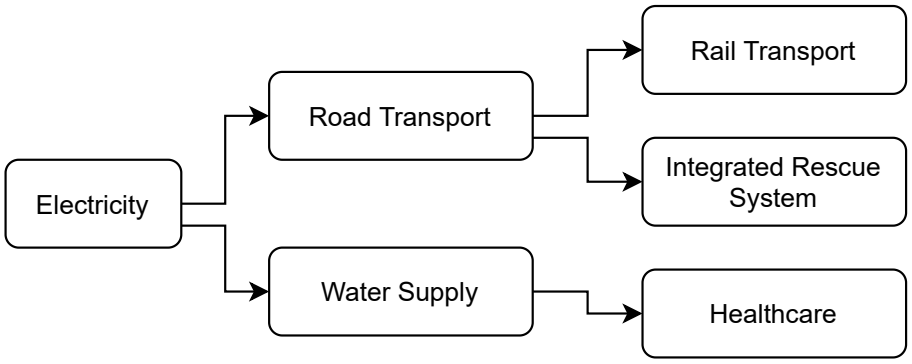


Figure 2.1: Links between sub-sectors for the case study in [23].

Table 2.1 presents the correlation matrix for this case study. For example, the entry (electricity, road transport) is 1, which means that the sub-sector road transport can fail if electricity fails. Indirect correlations must also be identified in this matrix. For example, in Figure 2.1 there is no arrow from electricity to rail transport, but there is one from electricity to road transport and another from road transport to rail transport, which means that the entry (electricity, rail transport) in the matrix must also be 1. Table 2.1 also presents the values of the activity and passivity coefficients for the case study in [23].

Utne et al. [24] proposes another method that uses a diagram containing the dependencies between different sub-sectors. According to it, the initiating event that is the cause of the cascading impacts needs to be identified, as well as the interdependencies between sub-sectors. This information must be

Table 2.1: Correlation matrix between sub-sectors and activity and passivity coefficients for the case study in [23].

	Electricity	Road transport	Water Supply	Rail Transport	Integrated Rescue System	Healthcare	K_A
Electricity	0	1	1	1	1	1	1.0
Road transport	0	0	1	1	1	1	0.8
Water Supply	1	0	0	0	0	1	0.4
Rail Transport	1	1	0	0	0	0	0.4
Integrated Rescue System	0	1	0	0	0	1	0.4
Healthcare	0	0	1	0	0	0	0.2
K_P	0.4	0.6	0.6	0.4	0.4	0.8	

assembled into a cascade diagram (as in Figure 2.2).

For each node in this diagram (initiating event and sub-sectors), the following parameters need to be determined:

- The frequency (f) of the initiating event is the number of times per year that the event is expected to occur.
- The conditional probability (p) must be assessed for all sub-sectors. It corresponds to the probability that a given sub-sector will become unavailable, given that the previous sub-sector has become unavailable.
- The extent (e) must be assessed for all leaf sub-sectors (those that do not have further dependent sub-sectors). It corresponds to the number of people that will be affected in case the sub-sector becomes unavailable.
- The duration (d) must also be assessed for each leaf sub-sector. It corresponds to the amount of time (in hours) that the sub-sector is expected to remain unavailable.

The process for calculating the risk of the initiating event starts by calculating the expected consequence (C_j) for each of the leaf nodes, j, according to Equation 2.14.

$$C_j = p_j \times e_j \times d_j \tag{2.14}$$

Then, for the nodes corresponding to the remaining sub-sectors (i), their consequence is calculated by Equation 2.15, where $\sum_j C_j$ corresponds to the sum of the consequences of the sub-sectors that depend on it.

$$C_i = p_i \times \sum_j C_j \tag{2.15}$$

Finally, the risk for the initiating event is given by Equation 2.16, where $\sum_k C_k$ corresponds to the sum of the consequences of the sub-sectors that are directly affected by the initiating event. The units of this parameter are person-hours per year.

$$R = f \times \sum_k C_k \tag{2.16}$$

Figure 2.2 shows an example of one such diagram, containing the parameters mentioned above. In this example, the initiating event is an electricity cable short circuit. The calculated risk of the initiating event is 13.5 person-hours per year.

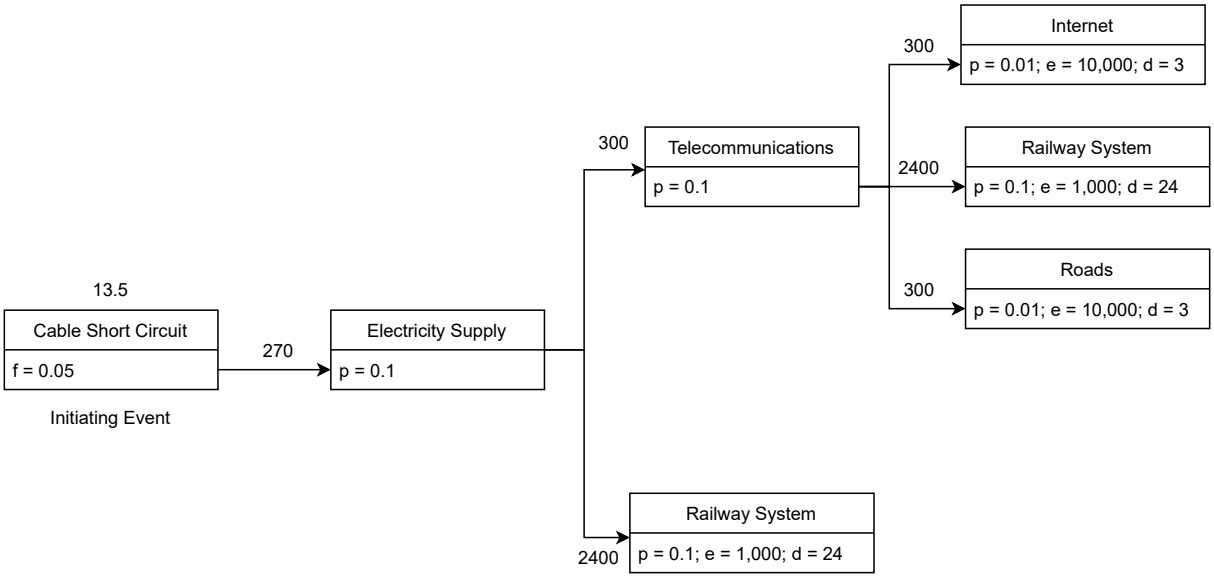


Figure 2.2: Cascade Diagram.

Haimes and Jiang [10] propose a method based on the Leontief Input-Output model [32]. This model was originally developed to study the equilibrium behavior of an economy, by modeling the dependencies between its various sectors. In [10], however, the model is adjusted to study the level of inoperability experienced by each infrastructure in a *Critical Infrastructure* system, due to *Cascading Failures*.

The method contains n infrastructures and the parameters at play are the following:

- x_j is the risk of inoperability of each infrastructure j. The concept of inoperability may have different meanings depending on the characteristics of the system being studied, but in general, it may take any value between 0 and 1, where 0 means the infrastructure is completely operable and 1 means the infrastructure is inoperable.

- a_{kj} is the probability of inoperability that the infrastructure j contributes to the infrastructure k. In other words, it is the degree of dependence of infrastructure k on infrastructure j. For example, if an infrastructure j fails and this leads to the certain failure of infrastructure k, then $a_{kj} = 1$. This parameter does not consider internal dependencies ($a_{kk} = 0$).

- x_{kj} is the level of inoperability that infrastructure j can trigger on infrastructure k, and is given by the following Equation:

$$x_{kj} = a_{kj} \times x_j \tag{2.17}$$

- c_k is the additional risk of inoperability caused by internal components of the infrastructure k, as well as by external perturbations, such that:

$$x_k = \sum_j x_{kj} + c_k = \sum_j a_{kj} x_j + c_k \tag{2.18}$$

Equation 2.18 can be written in matrix form as in Equation 2.19, where $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{c} = (c_1, \ldots, c_n)$ and A is a $n \times n$ matrix, where each entry A_{kj} is equal to a_{kj}.

$$\mathbf{x} = A\mathbf{x} + \mathbf{c} \tag{2.19}$$

Assuming matrix $(I - A)$ is non-singular, the solution of the problem is given by Equation 2.20.

$$\mathbf{x} = (I - A)^{-1}\mathbf{c} \tag{2.20}$$

For example, suppose this method is used to measure the *Cascading Failures* in a *Critical Infrastructure* system with two infrastructures and matrix A as in Equation 2.21, i.e., $a_{12} = 0.8$ and $a_{21} = 0.2$, which means that the degree of dependence of the first infrastructure on the second is 0.8, while the degree of dependence of the second infrastructure on the first is 0.2.

$$A = \begin{bmatrix} 0 & 0.8 \\ 0.2 & 0 \end{bmatrix} \tag{2.21}$$

If, for example, an attack occurs on the second infrastructure that makes it lose 60% of its functionality ($\mathbf{c} = (0, 0.6)$), then through Equation 2.20, the inoperability values of both infrastructures are calculated to be $\mathbf{x} = (0.571, 0.714)$.

In the method proposed by **Motter and Lai [25]** and further developed by **Wang et al. [16]** to include the concept of *betweenness*, the *Critical Infrastructure* entities (e.g., road segments and intersections in the Road transport sub-system; railway segments and railway stations in the Rail Transport sub-system; power lines in the Electricity sub-system) are represented by nodes in an undirected graph. There is an edge between two nodes whenever the real-life entities have links among themselves. For example, if a given road leads to a railway station and from that railway station leaves a set of rail tracks, then the node corresponding to the railway station would have edges to the road node, as well as to the railway track nodes.

Each node j in the graph has a capacity, C_j, given by Equation 2.22, where α is a tolerance parameter ($\alpha \geq 0$), and B_j is the *betweenness* of node j. The *betweenness* is a measure of the centrality of a node in the graph. It is calculated by finding all the shortest paths between every pair of nodes in the

graph and counting how many of them contain node j.

$$C_j = (1 + \alpha)B_j \tag{2.22}$$

When an attack occurs, the targeted k nodes are considered to fail and are removed from the graph. Then, the *betweenness* needs to be recalculated for each of the remaining nodes. If, for a node j, its recalculated *betweenness* is greater than its capacity ($B_j > C_j$), then node j also fails. This is considered a failure by overload since the new load on the node (here represented by its *betweenness*) surpasses the maximum supported load. Every failing node is removed from the graph and this process is repeated until no more nodes suffer failures. This whole process is very similar to the one proposed by the methods in Section 2.1.1, in the context of *Supply Chains*.

To estimate the impact of *Cascading Failures*, this method introduces the Fraction of Functional Entities (FF), calculated by Equation 2.23, where N is the total number of nodes and N_t is the number of remaining nodes at the end of the cascading failure process. The impact on the network is greater when the number of functional nodes is small, i.e., when FF is closer to 0.

$$FF = \frac{N_t}{N} \tag{2.23}$$

Duenas-Osorio and Vemuru [26] use this exact same method as [16, 25] but propose an alternative way of quantifying the impact of *Cascading Failures*. This method focuses on electric grid systems, so it differentiates the nodes between generation nodes and distribution nodes. It introduces the concept of connectivity loss (C_L), which is a measure of the average decrease in the number of generation nodes connected to distribution nodes, and is calculated using Equation 2.24, where n_D is the number of distribution nodes, n_G is the number of generation nodes and n_G^i is the number of generation nodes that have a path to the distribution node i.

$$C_L = 1 - \frac{1}{n_D} \sum_{i=1}^{n_D} \frac{n_G^i}{n_G} \tag{2.24}$$

To provide a measure of the impact of the *Cascading Failures*, it is necessary to compare the value of the connectivity loss before and after the failures occur. So, a new metric is defined — the Cascading Susceptibility (C_s). This parameter measures the percentage of added connectivity loss, after the *Cascading Failures*, in relation to the original value, before any failure takes place. It is calculated using Equation 2.25, where $C_{L,0}$ is the original value for the connectivity loss and $C_{L,Cascade}$ is the value for the connectivity loss after the occurrence of the *Cascading Failures* and the network stabilizes.

$$C_s = \frac{C_{L,0} - C_{L,Cascade}}{C_{L,0}} \tag{2.25}$$

The method developed by **Köpke et al. [27]** has the goal of improving Safety, Security and Resilience (SSR) in Offshore Wind Farms (OWF) infrastructure. To this end, it employs the FRAM model (Function

Resonance Analysis Method), which uses a representation of the system in terms of processes and functions, rather than the physical structure.

The first step is to identify the general SSR goals in the case study, by considering the different stakeholders (e.g., owners, operators, managers, etc.) and analysing their tasks and objectives in the OWF. In this case, the goals identified were accident prevention, security, compliance, occupational safety, environmental protection, reputation, plant safety, supply reliability, and finance. From these, it is necessary to select the more critical ones, by looking at the dependencies between them. The more critical goals are the ones that are most dependable (i.e., that have the most influence on the other goals). In this case, the most critical goals were accident prevention, security, occupational safety, plant safety, and environmental protection.

The next step consists on listing all the SSR functions. Each function belongs to one of the following categories:

- Detailed SSR goals — detailed functions that belong to the identified critical SSR goals — accident prevention (e.g., safety plane, safety ship, safety helicopter); security (e.g., protect cable, safe data, safe communication); occupational safety (e.g., safety of worker, shipwrecked men rescued); environmental protection (e.g., protect plants, protect water quality, protect animals); and plant safety (e.g., protect foundation, protect tower, protect converter station).
- Analysis of sensor data (e.g., observe water quality, heat detection, smoke detection, weather data).
- SSR measures — functions that are controlled and supervised by sensor data analysis (e.g., weather measures, fire detection, regular maintenance).

It is also necessary to identify the dependencies between functions, i.e., which functions does a given function need in order to operate properly. Given a function N_i, the functions that depend on it will be called its downstream functions, and the function that it depends on will be its upstream functions. Also, for each function identified, N_i, three parameters need to be assessed — the function's probability of failure (or not performing as expected), p_i, which can take the values Low (0.005), Medium (0.015) or High (0.02); the function's time to recover after a failure, t_i, which can be Low (2 to 5 days), Medium (6 to 30 days) or High (30 to 50 days); and the factor to influence downstream functions, f_i, which can be Low (1.05), Medium (1.2) or High (1.35).

Next, a simulation is performed based on the Monte Carlo method — for each day in a year ($n = 1, \ldots, 365$), and for each function N_i, the algorithm generates a random number $a_{n,i}$, between 0 and 1. This number will dictate whether function N_i will fail on day n (the function fails if $a_{n,i} < p_i$). In case this happens, the probabilities of failure of all downstream functions of N_i, $N_{i_{down}}$, are multiplied by the influence factor of N_i ($p_{i_{down}} := p_{i_{down}} \times f_i$). The failed function, N_i, is given a countdown time b_i, initially equal to t_i and decrements with each iteration of n. This parameter b_i represents the remaining time until the function is restored. To make sure that N_i does not fail again while it is already failed, its probability of failure, p_i, is temporarily set to 0. When function N_i is restored, i.e., when b_i reaches 0, the probabilities of failure of its downstream functions are also restored to the previous values ($p_{i_{down}} := p_{i_{down}} \div f_i$).

By repeating this algorithm multiple times, it is possible to, for example, obtain the probability distribution for the number of failures per year for a given function.

Ramirez Agudelo et al. [28] proposes a method that is used in the same context as [27] (i.e., improving/assessing Safety, Security and Resilience in Offshore Wind Farms, with the FRAM model), but instead of a Monte Carlo Simulations, it uses Bayesian Networks. The Bayesian Network is represented as a Directed and Acyclic Graph (DAG), where each node represents a function and an edge between two nodes represents a dependency between functions (for example, if a function B depends on function A, there is an edge from A to B). The goal is to calculate the probability of failure ($P(N_i)$) for each function N_i. For independent functions, i.e., functions that do not depend on any other function, this probability is simply the parameter p_i introduced in [27] ($P(N_i) = p_i$). As for dependent functions, the conditional probability of them failing, knowing the state of failure of its upstream functions is given by Equation 2.26, where $N_{i_{upstream}}$ is the set of M upstream function of N_i and F_j is given by Equation 2.27.

$$P(N_i|N_{i_{upstream}}) = p_i \prod_{j=1}^{M} F_j \tag{2.26}$$

$$F_j = \begin{cases} 1, & \text{if } N_j \text{ is not failed} \\ f_j, & \text{if } N_j \text{ is failed} \end{cases} \tag{2.27}$$

From the conditional probabilities calculated using Equation 2.26, it is then possible to calculate the actual probability of failure of each function N_i in the Bayesian Network ($P(N_i)$). Section 2.2.7 explains this process in detail.

2.2 Impact Assessment of Cyber-Threats

In order to understand the process behind a cyber-attack, Section 2.2.1 presents the intrusion kill chain, i.e., a sequence of tasks that need to be performed by an attacker in order to reach his/her goal, and Section 2.2.2 presents the MITRE ATT&CK Framework, which contains the possible tools and techniques that can be used to perform those tasks.

Since all cyber-attacks are performed by exploiting vulnerabilities in a network [33], in order to assess the impact of cyber-threats on computer networks, one must (1) assess the vulnerabilities that are exploited by the cyber-threat and (2) determine the impact propagation of those vulnerabilities throughout the network. This section presents two methods that can be used to assess vulnerabilities: CVSS scores (Section 2.2.3) and the DREAD model (Section 2.2.4). For determining the impact propagation throughout the network, either Risk Assessment Graphs (Section 2.2.5), the VASM model (Vulnerability-Asset-Service-Mission) (Section 2.2.6), or Attack Graphs (Section 2.2.7) can be used. Finally, Section 2.2.8 presents the Business Impact Assessment (BIA) methodology that uses both the VASM model and attack graphs.

2.2.1 Intrusion Kill Chain

Hutchins et al. [34] states that cyber-attacks occur in a specific order of phases, called the kill chain phases. It is called a chain because the intruder is required to execute each of the phases in order to reach its goal. The phases in the kill chain model are the following:

- Reconnaissance — Research, identification, and selection of targets (e.g., through Internet websites, mailing lists).
- Weaponization — Creation of a remote access malware that exploits a given vulnerability on the target machine/network. The malware is then delivered in a payload (e.g., PDF, Microsoft Office files).
- Delivery — Transmission of the created malware weapon to the target (e.g., via email attachments, websites, USB removable devices).
- Exploitation — The malware code is triggered, which leads to the exploitation of the vulnerability in the target machine/network.
- Installation — The malware installs a backdoor on the target system, which will allow the intruder to have persistent access to it.
- Command and Control — The compromised machine sends an outbound beacon in order to establish a channel between itself and the intruder. From here, the intruder gains ”hands-on keyboard” access to the target system.
- Actions on Objectives — Intruder has access to his/her original goal, which may include collecting or encrypting information on the victim, threatening the availability or integrity of the data, or simply using the target machine as an intermediary to hop laterally across the network.

2.2.2 MITRE ATT&CK Framework

MITRE’s ATT&CK Framework [35] describes how attackers operate, namely their goals and methods, based on real-world attacks. The framework contains, for each technology domain (Enterprise, Mobile, Industrial Control Systems), a set of tactics, that describe the goal of the intruder. For each tactic, the framework presents a list of techniques that describe how the intruder may try to achieve his/her objective. Each technique includes an ID, a description of the method used, the systems/platforms it applies to, and ways to mitigate and detect the technique being used.

Table 2.2 presents the tactics defined in MITRE’s ATT&CK Framework v8.2, with a brief description of the goal of the techniques belonging to each of the tactics and in which technology domains (Enterprise, Mobile, ICS) each tactic is defined. One relevant observation is that each of the tactics can be mapped onto one (or more) of the kill chain phases defined in Section 2.2.1 [36].

Each of the techniques defined in this framework can be categorized by their level of difficulty. This is especially useful when trying to assess the probability of a given technique being used in an attack. Smith [37] proposes such a categorization, where each technique is placed in one of the following levels:

Table 2.2: Description of MITRE ATT&CK v8.2 tactics, and domains in which they are defined.

Tactic	Enterprise	Mobile	ICS	Description
Reconnaissance	✓			Techniques used to gather information on the system, such as details of the organization, infrastructure or staff
Resource Development	✓			Techniques to create or compromise/steal resources that will be used in a later phase (e.g., purchasing domains, creating email accounts)
Initial Access	✓	✓	✓	Techniques used to get an initial leverage on the system (e.g., phishing)
Execution	✓	✓	✓	Techniques that run malicious code on the system (e.g., using a remote access tool to run shell scripts)
Persistence	✓	✓	✓	Techniques to allow intruders to keep access to the system, even after restart, change of credentials and other interruptions (e.g., adding startup code)
Privilege Escalation	✓	✓		Techniques used to gain higher-level permissions on the system
Defense Evasion	✓	✓	✓	Techniques that help the intruder remain undetected (e.g., disabling security software)
Credential Access	✓	✓		Techniques used by the attacker to steal account names and passwords (e.g., brute force)
Discovery	✓	✓	✓	Techniques to gain information about the internal system network (e.g., network sniffing)
Lateral Movement	✓	✓	✓	Techniques used for hopping through multiple machines in order to gain access to the objective (e.g., SSH, telnet)
Collection	✓	✓	✓	Techniques used to collect data relevant to the intruder's final goal (e.g., clipboard data, screen capture)
Command and Control	✓	✓	✓	Techniques to establish contact with compromised systems inside the network
Exfiltration	✓	✓		Techniques used to steal data without drawing suspicion, for example by encrypting or compacting
Inhibit Response Function			✓	Techniques used to interfere with the safeguards put in place in the system (e.g., suppression of alarms)
Impair Process Control			✓	Techniques used to manipulate physical processes (e.g., modifying parameters used to instruct ICS)
Impact	✓	✓	✓	Techniques to compromise the integrity or availability of the system or its data (e.g., making data inaccessible by encrypting it)
Network Effects		✓		Techniques to manipulate network traffic arriving or leaving the target device (e.g., jamming Wi-Fi signals)
Remote Service Effects		✓		Techniques that try to control the target device using remote services (e.g., Google Drive, Google Find My Phone)

- Level 0 — Techniques that are not exploitable on their own, but rather make use of other techniques (e.g., Graphical User Interfaces can be leveraged by other techniques to run malicious code).
- Level 1 — Techniques that are easy to exploit by almost anyone and do not require malware,

scripts, or other tools.

- Level 2 — Techniques that require additional steps, such as running scripts.
- Level 3 — Techniques that require some level of infrastructure to be able to exploit.
- Level 4 — Most difficult techniques, that require in-depth knowledge of the OS being exploited.

MITRE offers two tools that make use of this framework in order to study intrusions into cyber systems — in the ATT&CK Navigator tool [38], it is possible to make selections of techniques, filtered by technology domain (Enterprise, Mobile, ICS), platform (Linux, Android, Windows, etc.), software (type of tool or malware used in the exploit), and threat group (list of techniques that are often used by known hacker groups). It is also possible to overlap these selections in order to determine the most critical techniques that have the potential to be used in a given scenario. The second tool is CALDERA [39]. CALDERA is a simulation tool that simulates an attack on a given network using the techniques from the ATT&CK framework selected by the user.

2.2.3 Common Vulnerability Scoring System

CVSS [40] (Common Vulnerability Scoring System) is an open security standard for assessing the severity of vulnerabilities in computer systems ([33, 41–50]). It contains numerous metrics, belonging to three categories – Base Metrics (intrinsic characteristics of the vulnerability); Temporal Metrics (characteristics that evolve over time due to developments external to the vulnerability), and Environmental Metrics (customized to reflect the impact of the vulnerability on a given organization). The values for the metrics are usually qualitative but have a quantitative value associated.

NVD [51] (National Vulnerability Database) is a repository provided by NIST (National Institute of Standards and Technology) that contains security information, including CVSS scores, for a long list of known vulnerabilities. The CVSS Score provided by NVD corresponds to the Base Score, which is a value from 0 to 10 calculated only from the Base Metrics. Each vulnerability in this database also contains a CVE (Common Vulnerabilities and Exposures) Identifier, a brief description of the vulnerability, and all the values of the metrics used to calculate the Base Score. Table 2.3 contains the possible values for each of the Base Metrics and how they should be assigned, as defined in CVSS version 3.1 [52]. When assessing a vulnerability, each base score is placed in a category according to the descriptions in the third column of Table 2.3, and assigned a discrete value. For instance, the vulnerability CVE-2014-0160 (Heartbleed)[1], which allows remote attackers to obtain sensitive information from process memory, can be remotely exploited, which means that the Base Metric *Attack Vector* (AV) will be assigned a value of 0.85 (Network).

CVSS version 3.1 [52] also defines the expressions used to calculate the Base Score. From the Base Metrics, the Impact Sub-Score (ISS) is defined using Equation 2.28, which is then used to define the Impact according to Equation 2.29. The Exploitability is defined with Equation 2.30. Finally, the Base

[1] https://nvd.nist.gov/vuln/detail/cve-2014-0160

Table 2.3: Description and possible values of Base Metrics in CVSS v3.1 [52].

Metric	Description	Possible Values
Attack Vector (AV)	Context by which it is possible to exploit the vulnerability	**Network** (0.85) — The vulnerability is remotely exploitable, through the Internet **Adjacent** (0.62) — Attacker must have access to the physical (e.g., Bluetooth or IEEE 802.11) or logical (e.g., local IP subnet) network of the vulnerable asset **Local** (0.55) — Attacker must be able to access the vulnerable system either locally (e.g., keyboard) or remotely (e.g., SSH) **Physical** (0.2) — Attacker needs to be in physical contact with the vulnerable asset (e.g., inserting a USB stick)
Attack Complexity (AC)	Required conditions, beyond the attacker's control, that must be verified in order to exploit the vulnerability	**Low** (0.77) — No special conditions are required **High** (0.44) — The attack requires special conditions, such as only being possible in a limited time interval, or requires intimate knowledge such as configuration settings or sequence numbers
User Interaction (UI)	Level of requirement that a user participates in the exploitation of the vulnerability	**None** (0.85) — There is no need for a user to participate **Required** (0.62) — A user other than the attacker must take some action in order for the vulnerability to be exploited, for example by installing an application in the vulnerable system
Scope (S)	Defines whether the vulnerability can be exploited to impact assets that belong to a different security authority	**Unchanged** — The impacted asset and the vulnerable asset are either the same or belong to the same security scope **Changed** — The impacted asset is outside of security boundary of the vulnerable component (e.g., impacting the Operating System through a vulnerability in Google Chrome)
Privileges Required (PR)	Level of privileges required to exploit the vulnerability	**None** (0.85) — No special privileges required **Low** (0.62 if Scope Unchanged; 0.68 if Scope Changed) — Attack requires basic user privileges **High** (0.27 if Scope Unchanged; 0.50 if Scope Changed) — Attack requires administrative-level privileges
Confiden-tiality (C)	Level of loss of confidentiality — information access to only authorized users — suffered from the vulnerability	**High** (0.56) — Total loss of Confidentiality/Integrity/ Availability on the impacted asset **Low** (0.22) — Some loss of Confidentiality/Integrity/ Availability **None** (0) — No impact on the Confidentiality/Integrity/ Availability
Integrity (I)	Level of loss of integrity — correctness of information — suffered from the vulnerability	
Availability (A)	Level of loss of availability — accessibility of information — suffered from the vulnerability	

Score is calculated from these parameters using Equation 2.31, where function $Round(\cdot)$ returns the smallest number, with one decimal place, higher than or equal to its input.

$$ISS = 1 - [(1 - C) \times (1 - I) \times (1 - A)] \tag{2.28}$$

$$Impact = \begin{cases} 6.42 \times ISS, & \text{if Scope is Unchanged} \\ 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15}, & \text{if Scope is Changed} \end{cases} \tag{2.29}$$

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI \tag{2.30}$$

$$BaseScore = \begin{cases} 0, & \text{if } Impact \leq 0 \\ Round(Min[Impact + Exploitability], 10), & \text{if Scope is Unchanged} \\ Round(Min[1.08 \times (Impact + Exploitability)], 10), & \text{if Scope is Changed} \end{cases} \tag{2.31}$$

The vulnerability is further given a Qualitative Severity Rating based on its Base Score: None for scores of 0.0; Low for scores between 0.1 and 3.9; Medium for scores between 4.0 and 6.9; High for scores between 7.0 and 8.9; and Critical for scores between 9.0 and 10.0.

It is important to highlight that Vulnerabilities' CVSS Base Scores by themselves do not provide a good measure for the risk of a vulnerability. It is necessary to contextualize these threats and make an assessment that takes into consideration the specific case study environment [53].

2.2.4 DREAD Model

DREAD [54] is a threat modeling method that provides the level of risk of a given vulnerability. This model contains five categories of risk: Damage Potential — level of damage caused if the vulnerability is exploited; Reproducibility — ease of reproducing the vulnerability; Exploitability — ease of exploiting the vulnerability; Affected Users — quantity of users that will be affected by the exploitation of the vulnerability; and Discoverability — ease of discovering the vulnerability. Each vulnerability is given a rating of Low (1), Medium (2), or High (3) in each of these five categories. In the end, the five ratings are added, which results in a final risk rating. A vulnerability is considered high risk if it has a rating between 12 and 15; medium risk if the rating is between 8 and 11; and low risk if it is between 5 and 7.

This method has the advantage of being simpler than CVSS since it has fewer metrics to evaluate but has the disadvantages of being more subjective and of not existing a public database with ratings for known vulnerabilities.

2.2.5 Risk Assessment Graphs

Kheir et al. [45] introduces the concept of Risk Assessment Graph (RAG). The purpose of this graph is to estimate the risk of Information and Communications Technology (ICT) systems and how it evolves over time, based on their vulnerabilities and topology of the network.

Figure 2.3 presents an example of a Risk Assessment Graph (RAG). The triangular nodes u represent access points. In this context, access points are possible entry points for attacks. The rectangular

nodes $w = (a, v)$ represent asset-vulnerability pairs, where a is the asset and v is its vulnerability. Each node w has the associated values f_w^t and I_w.

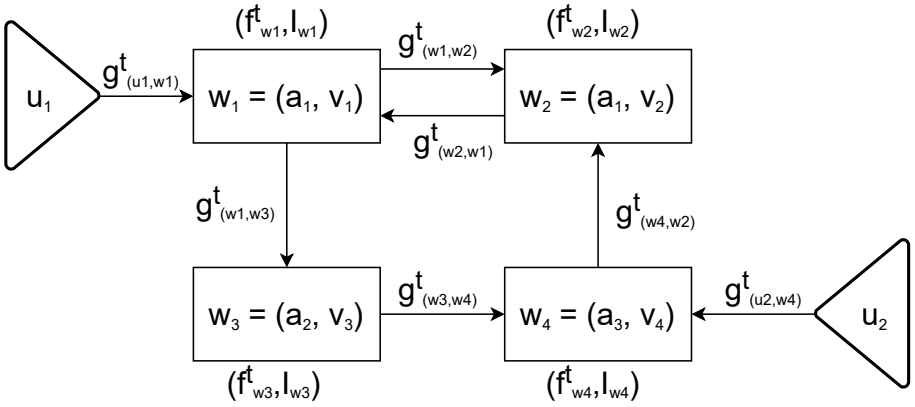


Figure 2.3: Risk Assessment Graph (RAG).

f_w^t is the potentiality function – it measures the likelihood of vulnerability w being directly exploited at least once before time t. It is a function that depends on t and can be estimated by the Equation 2.32, where α_w is a parameter between 0 and 1 that controls how fast the potentiality of node w converges to 1, and p_w is the probability of exploitation of node w, which can be acquired from the NVD-CVSS database.

$$f_w^t = 1 - (1 - p_w)^{\alpha_w \times t} \tag{2.32}$$

I_w is the impact of node $w = (v, a)$. In other words, it is the level of damage generated by exploiting vulnerability v on asset a. This parameter also comes from the NVD-CVSS database.

Whenever the exploitation of w_i makes the exploitation of w_j possible, there is an arc from w_i to w_j. This arc has a weight $g_{(w_i,w_j)}^t$. This weight is called the accessibility function and measures the frequency of access between the assets of w_i and w_j. If w_i and w_j correspond to the same asset (like w_1 and w_2 in Figure 2.3), then the accessibility function of their arc will be 1 ($g_{(w_i,w_j)}^t = 1$).

The propagation function between two nodes w_i and w_j, $h_{(w_i,w_j)}^t$, will be a value between 0 and 1 defined as the product between the potentiality function of the destination node w_j and the accessibility function of the arc between the two nodes, as shown in Equation 2.33.

$$h_{(w_i,w_j)}^t = f_{w_j}^t \times g_{(w_i,w_j)}^t \tag{2.33}$$

Let π be a path with length k from the access point u to the node w. The propagated potentiality on path π, $P_{u,w}^{\pi,t}$, is defined in Equation 2.34 as the product of the propagation functions of every arc in π.

$$P_{u,w}^{\pi,t} = \prod_{i=1}^{k-1} h_{(w_i,w_{i+1})}^t \tag{2.34}$$

The most likely path between u and w at time t will be the path with the maximum propagated potentiality, and its potentiality is $P_{u,w}^t = \max_{\pi} P_{u,w}^{\pi,t}$. This is the only potentiality value needed, since it is

the upper bound for the potentialities of all paths between u and v.

Finally, the risk of each node can be calculated, as well as the global risk. The first step is to calculate the propagated risk from access point u to node w at time t, $R^t_{u,w}$, according to Equation 2.35. The risk of each node w, R^t_w, is the sum of propagated risks from u to w over all access points u (2.36), and the global risk, R^t, is the sum of the node risk of every node w (2.37).

$$R^t_{u,w} = P^t_{u,w} \times I_w \tag{2.35}$$

$$R^t_w = \sum_u R^t_{u,w} \tag{2.36}$$

$$R^t = \sum_w R^t_w \tag{2.37}$$

2.2.6 VASM model

The VASM model [46] (Vulnerability-Asset-Service-Mission) is a model that represents the interactions and dependencies between entities of an organization, through an Entity Dependency Graph, divided into layers. This model can be used to estimate the propagation of the impact of an attack on the organization's missions ([42, 46]).

The VASM model contains four layers, as presented in Figure 2.4. The mission layer contains the missions on which ultimately the impact of attacks will be assessed (green circles), as well as the tasks that constitute each mission (red circles). There are also two special nodes: AND-nodes and OR-nodes. When a mission OR-depends on several tasks, it means only one of those tasks needs to be complete for the mission to be complete. On the other hand, if a mission AND-depends on several tasks, all those tasks need to be complete in order for the mission to be complete. Each of these tasks can further depend on other tasks through these relationships. Each task may depend on one or several services, in the service layer. Examples of Services (blue circles) include Database, File Transfer, and E-mail. The dependencies between tasks and services can be through direct relationships (e.g., S1 to T1), OR-relationships (e.g., S2 OR S4 to T2), and AND-relationships (e.g., S3 AND S6 to T4). A direct relationship means the mission only depends on its predecessor service (although this service may depend on other services); an OR-relationship means a task can be completed as long as any one of its prerequisite services is available; and an AND-relationship means that the completion of the task requires the availability of all its predecessor services. In the service layer, each service can also depend on one or multiple assets (in the asset layer) through direct, AND, or OR-relationships, and a service can also depend on other services through direct relationships. The asset layer includes software (e.g., OS, Middleware, Application) and hardware (e.g., Routers, Servers, Firewalls, Sensors, Printers) assets, represented by orange circles. Each asset can have one or more vulnerabilities (in the vulnerability layer) and each vulnerability can be in more than one asset. Vulnerabilities (purple circles) can also have dependencies between themselves – when this happens, the attacker must exploit each

vulnerability in the dependency in that specific order.

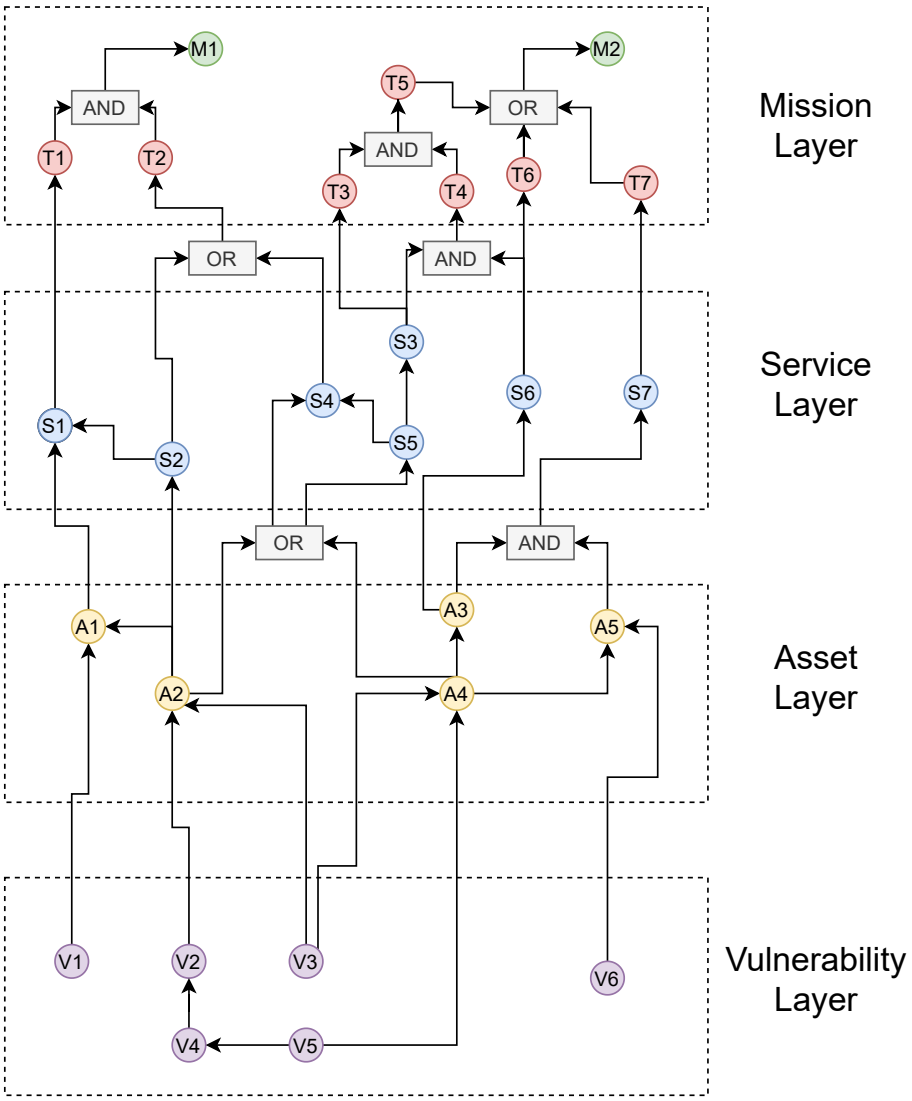


Figure 2.4: VASM model.

The method for assessing the impact propagation in the VASM, proposed by **Sun et al. [46]**, defines two parameters — Operational Capacity (OC) and Impact Factor (IF). The operational capacity (OC) is a parameter of all Mission, Task, Service, and Asset nodes. It is a value between 0 and 1 that measures the execution ability of a node (a value of 1 means the node is fully operational and a value of 0 means the node is inoperable). The Impact Factor (IF) is a parameter relative to the Vulnerability nodes, that measures to what degree the vulnerability is capable of compromising the attacked asset, on a scale from 0 to 1.

The first step of this method is to assess the probability (P) and vulnerability level (V) of each vulnerability in order to calculate its impact factor (IF). The vulnerability level is determined from the vulnerability's CVSS score. Since this score is a value between 0 and 10 and the intended value is between 0 and 1, the vulnerability level of vulnerability i is simply given by the Equation $V_i = \frac{CVSS_i}{10}$.

The probability of occurrence of an attack that exploits a given vulnerability i, P_i, depends on the Attack Complexity (level of difficulty in the implementation of the attack) and Attack Timeliness (the time

it takes for the attack to cause significant impact – the shorter the time, the higher the timeliness). In general, the probability of the attack is greater when its complexity is low, and its timeliness is high. Table 2.4 presents the attack's probability in function of its complexity and timeliness. To convert the table into numbers between 0 and 1, it is considered that a probability of High is 0.9; Medium is 0.5 and Low is 0.2.

Table 2.4: Attack Probability, depending on Attack Timeliness and Attack Complexity.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">Attack Complexity</td></tr>
<tr><td>High</td><td>Medium</td><td>Low</td></tr>
<tr><td rowspan="3">Attack Timeliness</td><td>High</td><td rowspan="2"></td><td rowspan="2">Medium</td><td rowspan="2">High</td></tr>
<tr><td>Medium</td></tr>
<tr><td>Low</td><td colspan="3">Low</td></tr>
</table>

The Attack Complexity can in turn be determined by the Level of Defensive Measures installed in the target system and by the Study Level of the Vulnerabilities (Level of knowledge required to successfully exploit the vulnerability – the higher the Study Level, the easier it is to exploit). The estimated Attack Complexity based on these two factors can be obtained by accessing Table 2.5.

Table 2.5: Attack Complexity, depending on Level of Defensive Measures and Study Level of the Vulnerability.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="3">Study Level of the Vulnerability</td></tr>
<tr><td>High</td><td>Medium</td><td>Low</td></tr>
<tr><td rowspan="3">Level of Defensive Measures</td><td>High</td><td colspan="2"></td><td>High</td></tr>
<tr><td>Medium</td><td rowspan="2"></td><td>Medium</td><td rowspan="2"></td></tr>
<tr><td>Low</td><td colspan="2">Low</td></tr>
</table>

The Study Level of the Vulnerability, Level of Defensive Measures and Attack Timeliness need to be assessed for each vulnerability in order to calculate the Probability of Attack. With the Probability of Attack (P_i) and level of vulnerability of i (V_i), it is possible to calculate the impact factor of vulnerability i, using Equation 2.38.

$$IF_i := V_i \times P_i \tag{2.38}$$

In case there is a dependency between vulnerabilities (vulnerability i depends on vulnerability j), then the impact factor must be updated to $IF_i := max(IF_i; IF_j)$.

For each asset in the asset layer, the total impact factor must be calculated. If an asset has n vulnerabilities and the attacker has managed to exploit k of them, then the total impact factor of the asset is given by Equation 2.39.

$$IF_{total} = max\left(IF_{i;i\epsilon(1,k)}\right) + \left(1 - max\left(IF_{i;i\epsilon(1,k)}\right)\right) \times \frac{\sum_{j=1}^{k} IF_j - max\left(IF_{i;i\epsilon(1,k)}\right)}{\sum_{j=1}^{n} IF_j - max\left(IF_{i;i\epsilon(1,k)}\right)} \tag{2.39}$$

If the asset has all its vulnerabilities exploited ($n = k$), then the Equation 2.39 results in $IF_{total} = 1$.

After an attack, the operational capacity of an asset x, OC_x^A, is updated according to Equation 2.40.

$$OC_x^A := OC_x^A \times (1 - IF_{total,x}) \tag{2.40}$$

In case asset x depends on another asset, y, its OC is further updated to $OC_x^A := min(OC_x^A, OC_y^A)$.

The OC of services is updated as follows:

- If the service has a direct dependency on an asset, then the OC of the service will be the same as the asset's: $OC^S := OC^A$.
- If the service AND-depends on several assets, its OC will be the product of the assets' OCs: $OC^S := \prod_i OC_i^A$.
- If the service OR-depends on several assets, its OC will be the average of the assets' OCs: $OC^S := avg(OC_1^A, \ldots, OC_n^A)$.

Like before, if a service depends on another service, its OC is further updated like so: $OC_x^S := min(OC_x^S, OC_y^S)$.

The update of tasks' OC based on services' OC follows the same rules stated above for the update of services OC based on assets OC.

The OC of a mission will be the minimum between the OCs of the tasks that belong to the mission and were actually executed. For example, if a mission OR-depends on several tasks, only one of those tasks will actually be executed, so the mission will take that task's OC ($OC_{OR}^M := OC^T$). However, if a mission AND-depends on several tasks, all those tasks will have to be executed, so the mission's OC will be the minimum between the OCs of those tasks ($OC^M := min(OC_1^T, \ldots, OC_n^T)$).

Jakobson [42] proposes a simpler impact propagation method that still uses the VASM model, Impact Factors, and Operations Capacities. According to this method, each Vulnerability i has an Impact Factor (IF_i) that directly results from its CVSS score, as in Equation 2.41.

$$IF_i := \frac{CVSS_i}{10} \tag{2.41}$$

If an asset a has a vulnerability x and does not depend on any other asset, its OC will be updated as in Equation. 2.42.

$$OC_a^A := max(OC_a^A - IF_x; 0) \tag{2.42}$$

If, however, this asset a does depend on another asset b, its OC will be updated according to Equation 2.43.

$$OC_a^A := min(max(OC_a^A - IF_x; 0); OC_b^A) \tag{2.43}$$

For the propagation of the OC to the Service and Task nodes, there can be three situations:

- Propagation through a direct dependency. In this case the parent node will take its child's OC: $OC_{parent} := OC_{child}$.
- Propagation through an AND-node. In this case, the OC of the parent will be the minimum between the OCs of its children: $OC_{AND} := min(OC_1, \ldots, OC_n)$.
- Propagation through an OR-node. In this case the OC of the parent will be the average between the OCs of its children: $OC_{OR} := avg(OC_1, \ldots, OC_n)$.

Finally, the OC of a mission will be the product of the OCs of its tasks that have been executed ($OC^M := OC_1^T \times \ldots \times OC_n^T$).

2.2.7 Attack Graphs

An attack graph is a representation of possible attacks against a given network. This type of graph can be constructed by a software tool called MulVAL (Multi-host, Multi-stage Vulnerability Analysis Language) [55] – MulVAL builds the attack graph based on the IT network topology and other characteristics.

Figure 2.5 presents an example of an attack graph produced by MulVAL. It contains three kinds of vertices: rectangle vertices (SINK) represent ground facts, which includes vulnerabilities of each asset, services running on each machine, and connectivity between assets; elliptic vertices (AND) represent reasoning rules, that define how a privilege may be achieved; and diamond vertices (OR) represent derived attack assets, i.e., privileges an attacker can obtain by exploiting the vulnerabilities in the system [56]. In general, the root of the graph is a diamond (OR-node) that represents the ultimate attacker goal, the leaves are rectangle nodes (SINKs) that represent facts about the system, and the intermediate nodes are steps the attacker needs to perform to reach his/her goal [57].

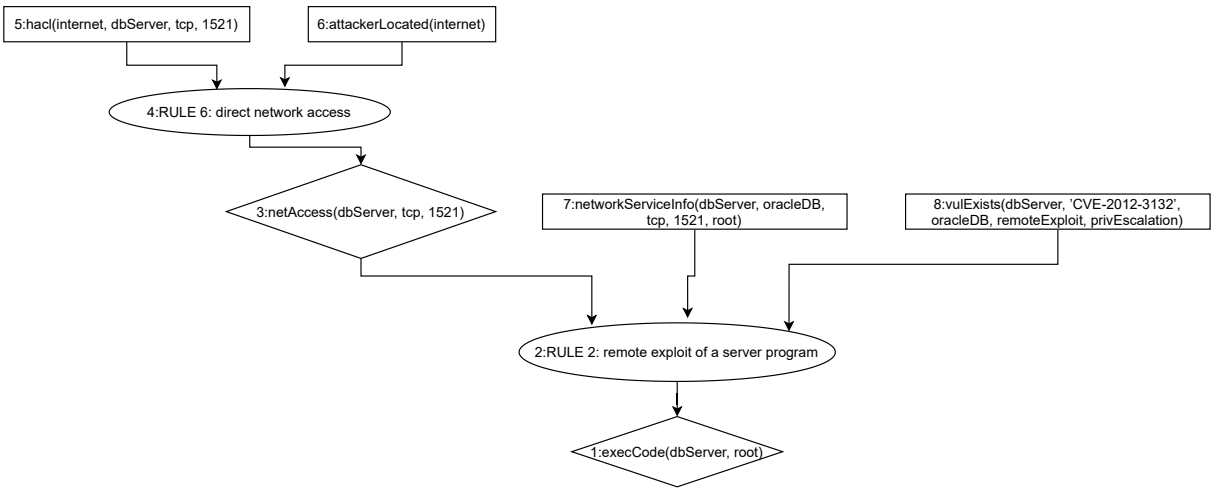


Figure 2.5: Example of an Attack Graph, from [57].

The problem with attack graphs is that they can be very complex and thus very hard for a human to understand the security problems of the network. This section analyses algorithms that make this process more automated, by computing the level of criticality [41] (also called the level of impact [33, 47], level of vulnerability [58], or probability of compromise [48–50]) for each node in the attack graph.

An important distinction in the presented methods is between logic-based models and bayesian-based models. The logic-based models ([33, 41, 47, 58]), use a sequential process in order to determine the level of criticality for each node, while the bayesian-based models ([48–50]) have the advantage of being able to calculate not only the unconditional probabilities of compromise for each node, but also the probabilities of compromise based on *a posteriori* information, i.e., they can calculate the probabilities of each node being compromised supposing that the node corresponding to the attacker's goal has been compromised.

Sawilla and Ou [41] propose an algorithm that can help condense the vast amount of information into a list of priorities, by assigning a rank of criticality to each vulnerability. This algorithm uses not only dependency relationships in the attack graph but also attributes of the security problems, not present in the graph. The algorithm consists on solving Equation 2.44, which is equivalent to finding the primary eigenvector of matrix $(D\Delta + \gamma P 1^T)$.

$$\lambda X = (D\Delta + \gamma P 1^T)X \tag{2.44}$$

In Equation 2.44:

- λ is the eigenvalue of the equation.
- X is the eigenvector and corresponds to the intended rank vector. It is a $n \times 1$ vector where n is the number of vertices of the graph. The rank of each vertex represents the criticality of the corresponding asset. X must be normalized so that its elements sum to 1.
- D is a $n \times n$ matrix that is the transpose of the graph's edge weights with normalized rows. In the graph, each edge (u, v) (pointing from u to v) has a weight $g(u, v)$ that is given by Equation 2.45. Since this weight does not depend on the vertex u, i.e., it is only determined by vertex v, then it can also be represented by $m(v)$.

$$g(u,v) = m(v) = \begin{cases} s(v), & \text{if } v \text{ is SINK vertex} \\ s(v) \times \prod\limits_{w \epsilon N^+(v)} m(w), & \text{if } v \text{ is AND vertex} \\ \max\limits_{w \epsilon N^+(v)} m(w), & \text{if } v \text{ is OR vertex} \end{cases} \tag{2.45}$$

(Note: $w \epsilon N^+(v)$ means that w in an out-neighbour of v, meaning edge vw exists in the graph).

Matrix D is the transpose of g' ($D = g'^T$), where g' is the normalization of g according to Equation 2.46.

$$\sum_{w \epsilon N^+(v)} g'(v,w) = \begin{cases} |N^+(v)|, & \text{if } v \text{ is AND vertex} \\ 1, & \text{if } v \text{ is OR vertex} \\ 0, & \text{if } v \text{ is SINK vertex} \end{cases} \tag{2.46}$$

In Equation 2.45, $s(v)$ is the success likelihood of vertex v. It can be interpreted as the ease of exploiting vertex v, where a value of 0 means that it is very difficult to exploit and a value of 1 means

that it is very easy. This parameter is subject to a certain level of subjectivity. One systematic way of assigning it is as follows:

  - SINK vertices can either represent asset vulnerabilities or some other ground fact (for example, connectivity between assets and services running on machines). For SINK vertices that do not correspond to vulnerabilities, $s(v)$ is assigned the probability of the service/network being up (by default $s(v) = 1$). If a SINK vertex does correspond to a vulnerability, then $s(v)$ can be assigned according to public known information about the vulnerability — for example, $s(v)$ can be assigned to the CVSS score of the vulnerability ($s(v) = \frac{CVSS_v}{10}$).
  - For AND vertices (rules in MulVAL), $s(v)$ will be assigned according to the preference of attackers to different attack strategies. For example, it can be assigned $s(v) = 1$ for rules describing direct routes and $s(v) = 0.5$ for rules describing multi-hop access, since an attacker will prefer direct routes.
  - For OR vertices (derived assets), $s(v)$ represents the success likelihood of obtaining the derived asset using an "out-of-bands" attack, i.e., an attack that is not contemplated in the graph, such as social engineering attacks.

- Δ is a diagonal $n \times n$ matrix, where $\Delta_{v,v} = \delta_v$. The parameter δ_v measures the likelihood that an attacker will continue traversing the graph in order to reach vertex v. This parameter really only makes sense for OR vertices (derived assets) — the probability that an attacker will continue to follow the graph in order to reach a given OR vertex is greater if that vertex is hard to reach through "out-of-band" attacks (has a low $s(v)$) and easy to reach through the graph (has a high $m(v)$). To model this behavior, δ_v is assigned to OR vertices according to Equation 2.47. For AND and SINK vertices, $\delta_v = 1$.

$$\delta_{v(OR)} = (1 - s(v)) + s(v)m(v) \tag{2.47}$$

- P is a $n \times 1$ vector called the personalization vector. Each element represents the value of a vertex to an attacker. This vector should be 0 for every vertex except for the ones that correspond to the goal of the attacker (usually the root vertex).

- γ is a scalar parameter between 0 and 1. It determines the amount of importance that should be given to the attacker's goal (the closer γ is to 1, the more important is the goal).

- 1^T is a $1 \times n$ vector where all elements are 1.

Equation 2.44 can be solved iteratively by successively applying the steps in Equation 2.48, where t increments by 1 with each iteration.

$$\text{Step 1: } X'_t = D\Delta X_{t-1} + \gamma P \qquad \text{Step 2: } X_t = \frac{1}{||X'_t||_1} X'_t \tag{2.48}$$

The method proposed by **Cao et al. [33]** for estimating the impact of cyber attacks using attack

graphs is much less exhaustive. According to this method, each vertex i has an impact score V_i, between 0 and 1. For the vulnerability nodes (SINK), the impact is assessed based on their CVSS scores ($V_i = \frac{CVSS_i}{10}$). For AND/OR-nodes that depend on nodes i and j, their impact is given by Equations 2.49 and 2.50, respectively.

$$V_{AND} = V_i \times V_j \tag{2.49}$$

$$V_{OR} = V_i + V_j - V_i \times V_j \tag{2.50}$$

The impact of each node should be calculated for all nodes, from the leaves to the root.

Noel et al. [47] proposes a method that uses these same rules for impact propagation in AND and OR nodes as Cao et al. [33]. However, the impact measure of each vertex is not deterministic but rather follows a probability distribution. This probability distribution must be specified for each of the leaf vertices (SINKs). For example, it can be a uniform distribution between $CVSS_i - \epsilon$ and $CVSS_i + \epsilon$, where $CVSS_i$ is the CVSS score of the vulnerability. Then, the Monte Carlo method is used to calculate the probability distribution of the impact measure for all the other vertices. This method consists on generating random values of impact for the leaf vertices (according to their probability distribution) and then calculating the impact measure of all the other vertices using the propagation laws in Equations 2.49 and 2.50. This process is repeated many times in order to obtain, for each vertex, an approximation of its impact measure probability distribution.

The method proposed by **Ten et al. [58]** does not compute an individual impact score for each node in the graph. Instead, it considers all possible scenarios that can lead to the exploitation of the root node and produces a vulnerability score for the whole system. First, each leaf node is assigned a vulnerability index based on (1) the existing countermeasures in place to counteract the vulnerability, (2) the history of attempted intrusions using that vulnerability, and (3) the password policy enforcement. For each of the leaf nodes, the three following conditions are assessed:

- There is no history of intrusion attempts using the vulnerability.
- The vulnerability is protected by one or more countermeasures.
- One or more password policies are enforced corresponding to the vulnerability.

From these conditions, the cybersecurity condition parameter, ω_k, is assessed for every leaf node k — if all conditions are met, then $\omega_k = 0$; if any two conditions are met, then $\omega_k = 0.5$; and if only one or no conditions are met, then $\omega_k = 1$. With this parameter, the vulnerability of each leaf node k, $V(G_k)$, is computed using Equation 2.51, where n_k is the number of countermeasures implemented in the leaf node, and ρ_k is the weight of the password policy enforcement. This is a value between 0 and 1, where 1 means that no password policies are enforced (e.g., there is a guest account known to everyone, where the password is the same as the username) and 0 corresponds to exhaustive password policies (e.g., all passwords contain lower and upper case letters, numbers and non-alphanumeric symbols, have more than 8 characters and are required to be changed periodically).

$$V(G_k) = \begin{cases} max\{\omega_k(1 - \frac{n_k}{5}); \omega_k\rho_k\} & \text{, if } \omega_k > 0 \\ \frac{max\{1-\frac{n_k}{5}; \rho_k\}}{3} & \text{, if } \omega_k = 0 \end{cases} \tag{2.51}$$

From the vulnerability of the leaf nodes, it is possible to calculate the vulnerability index of a scenario. A scenario corresponds to a set of leaf nodes that, if exploited, enable the attacker to gain access to the root node. Also, a scenario does not contain redundant leaf nodes, i.e., all nodes in a scenario are required to exploit in order to access the root node. For example, in Figure 2.6, the possible scenarios are $\{1\}$, $\{2\}$ and $\{3,4\}$.

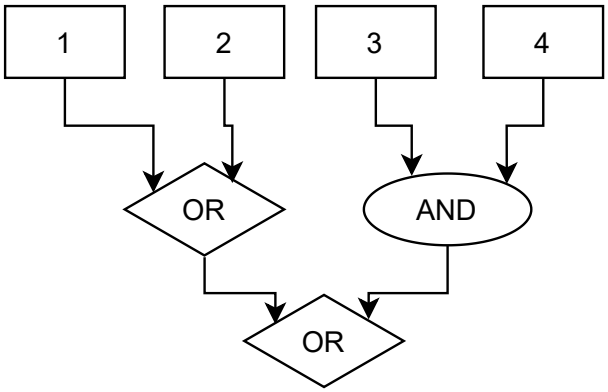


Figure 2.6: Attack Graph with four leaf nodes.

The vulnerability index of a scenario S_j, $V(S_j)$, is given by the product of the vulnerability indexes of all its leaf nodes, as shown in Equation 2.52.

$$V(S_j) = \prod_{k \epsilon S_j} V(G_k) \tag{2.52}$$

Finally, the vulnerability index of the whole system (V_s) is given by the maximum of the vulnerability indexes of all its possible scenarios:

$$V_s = \max_{S_j}\{V(S_j)\} \tag{2.53}$$

Another possible approach is applying the concept of *Bayesian Networks* to Attack Graphs **([48–50])**. According to this method, each vertex V is either compromised (with probability $P(V)$) or not compromised (with probability $P(\neg V) = 1 - P(V)$). The goal is to calculate these probabilities for every vertex.

For each of the leaf vertices (vulnerabilities), it is necessary to assign a prior probability of that vulnerability being compromised. This value can be estimated by each vulnerability's CVSS score. Each edge $V \rightarrow W$ in the graph also needs to be assigned a probability ($P(V \rightarrow W)$), corresponding to the probability that a vertex V is able to use that edge to compromise vertex W. One possible way to assign these values is through Equation 2.45. Figure 2.7 presents an example of an attack graph with probabilities for the leaves and the edges.

The conditional probability of a vertex V being compromised, knowing the state of compromise of its

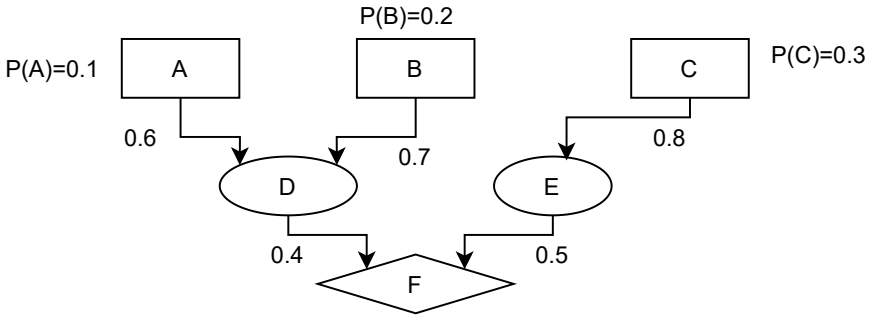


Figure 2.7: Example of a Bayesian Network in an Attack Graph.

parents ($Pa[V]$) is given by Equation 2.54 for AND vertices, and Equation 2.55 for OR vertices.

$$P(V_{AND}|Pa[V]) = \begin{cases} 0 & \text{, if at least one parent is not compromised} \\ \prod\limits_{Pa[V]_i} P(Pa[V]_i \rightarrow V) & \text{, otherwise} \end{cases} \tag{2.54}$$

$$P(V_{OR}|Pa[V]) = \begin{cases} 0 & \text{, if none of the parents are compromised} \\ 1 - \prod\limits_{Pa[V]_i} [1 - P(Pa[V]_i \rightarrow V)] & \text{, otherwise} \end{cases} \tag{2.55}$$

The probabilities of the remaining vertices V will be calculated by using the conditional probability formula (Equation 2.56) and summing over all possible values for states of compromise of the parents.

$$P(Y \cap X) = P(X) \times P(Y|X) \tag{2.56}$$

For example, in Figure 2.7, the probability of vertex D being compromised is given by Equation 2.57.

$$P(D) = P(D \cap (A \cap B)) + P(D \cap (\neg A \cap B)) + P(D \cap (A \cap \neg B)) + P(D \cap (\neg A \cap \neg B)) \tag{2.57}$$

By applying the conditional probability formula (Equation 2.56), and the Equation for the conditional probability of an AND vertex (2.54), the last three parcels become 0 which results in Equation 2.58.

$$\begin{aligned} P(D) &= P(D|(A \cap B)) \times P(A \cap B) \\ &= P(A \rightarrow D) \times P(B \rightarrow D) \times P(A) \times P(B). \\ &= 0.6 \times 0.7 \times 0.1 \times 0.2 = 8.4 \times 10^{-3} \end{aligned} \tag{2.58}$$

The same process can be applied to vertex E, which results in $P(E) = 0.3 \times 0.8 = 0.24$. And since vertex F is an OR vertex, Equation 2.55 should be used instead of Equation 2.54. The calculation of the

probability of F is shown in Equation 2.59.

$$\begin{aligned} P(F) &= P(F|(D \cap E)) \times P(D \cap E) + P(F|(\neg D \cap E)) \times P(\neg D \cap E) + P(F|(D \cap \neg E)) \times P(D \cap \neg E) \\ &= [1 - (1 - 0.4)(1 - 0.5)]P(D)P(E) + (0.5)(1 - P(D))P(E) + (0.4)P(D)(1 - P(E)) \\ &= 0.123 \end{aligned} \tag{2.59}$$

Assuming node F is the goal of the attacker in this scenario, $P(F)$ represents the probability that the attack will be successful.

This method can also be used to hypothesize an attack on a given vertex and calculate the *a posteriori* probabilities of each node, using Bayes' Theorem (Equation 2.60).

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \tag{2.60}$$

For example, suppose an attack occurred on vertex F. The *a posteriori* probability of E ($P(E|F)$) is given by $P(E|F) = P(F|E)P(E)/P(F) = 0.98$, where $P(F|E) = P(F|(E \cap D))P(D) + P(F|(E \cap \neg D))P(\neg D) = 0.502$. The *a priori* probability of E being compromised is 0.123, and the *a posteriori* probability of E being compromised, given that F was compromised, rose to 0.98. This method can be applied to all other vertices to determine how their probabilities change.

2.2.8 Business Impact Assessment

BIA (Business Impact Assessment) [11, 12] is a software tool developed by INOV (*Instituto de Engenharia de Sistemas e Computadores Inovação*) used to analyse the propagation of cyber threats in business organizations and determine their impact on the business's goals. It does so in three stages and uses a version of the VASM model in conjunction with attack graphs.

A layered model is used to describe all the relationships between the different entities that comprise the business network (e.g., which threats affect which assets (i.e., devices); which assets run which services; which services provide which activities, and which activities support which business-processes). Attack graphs are used in the context of MulVAL — based on a set of given rules and preconditions, MulVAL automatically generates an attack graph, from which the attack propagation paths will be obtained.

The first stage of BIA (Setup Stage) is used to build two models that capture (1) the organization network topology and (2) the business logic. It is essentially a specification-based system, however, part of the configuration can also be obtained based on observation of packet captures and firewall configuration files. The network model provides a list of assets and the connectivity between them. Each individual asset is then assigned to a type (e.g., PLC, SCADA, Access Point, etc.) and each type of asset is assigned to a set of threats that it is affected by (e.g., <Access Point, Man-in-the-Middle Attack>, <SCADA, Communication hijack>). Furthermore, in the business logic model, each asset is also mapped to one (or more) business-processes, which correspond to the services and mission tasks of the layered model.

An example of the layered model used by BIA is illustrated in Figure 2.8. The model is composed by the Asset, Service, and Activity Layers.

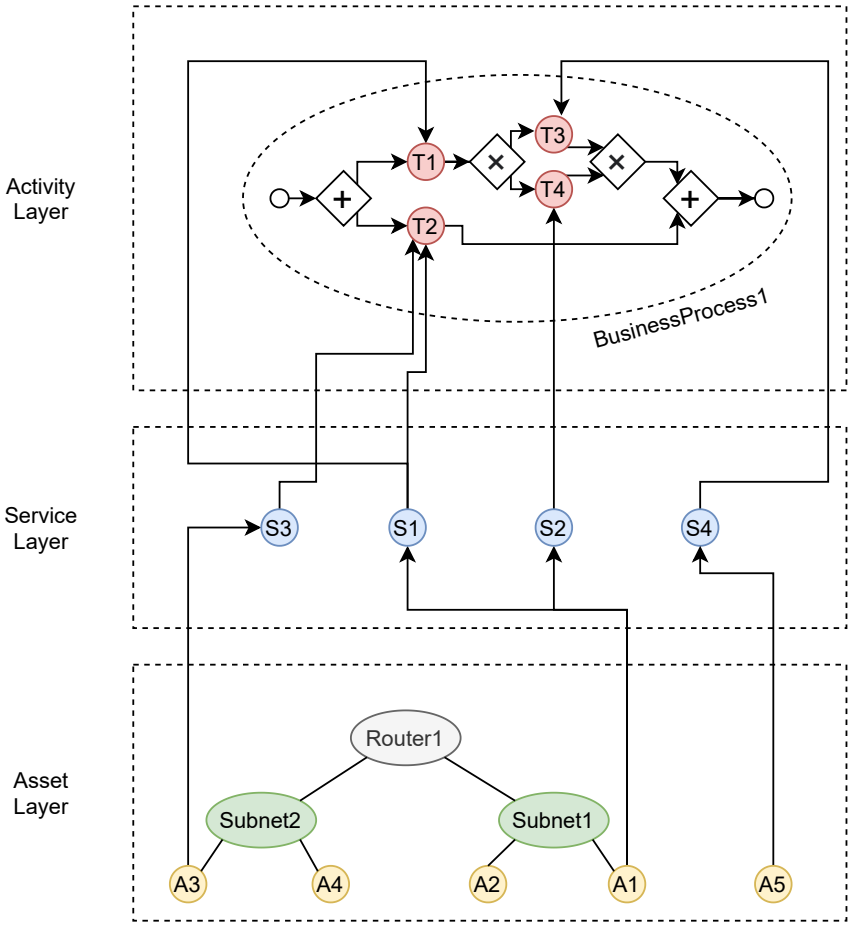


Figure 2.8: Example of network model used by BIA.

In the Asset Layer are represented physical devices of the network (yellow circles). Each of these assets contains a set of threats that can be exploited. The Asset Layer also models the connectivity between different assets by allowing each asset to belong to a subnet (green circles) and routers (grey circles) to establish communication between different subnets.

In the Service Layer, each service (blue circles) (e.g., Operating System, Middleware, Applications), is carried out by one or more assets. It is assumed, for simplicity, that if a service is provided by two different assets, each of the assets provides the whole service, such that the multiple assets have the purpose of redundancy. This corresponds to an OR-relationship in the VASM model (Section 2.2.6).

The topmost layer — Activity Layer — contains the business-processes and corresponding activities (red circles). An activity corresponds to an action that is carried in the context of a business-process. Like before, an activity can be provided by one or more services, and a single service provides the activity entirely. A business-process is defined as a sequence of activities with a start and an end, and can be modelled through a Business Processes Modelling Notation (BPMN) Diagram [59], depicted in Figure 2.9.

Three types of nodes are defined in the business-process diagram — *parallel gateways*, *inclusive*

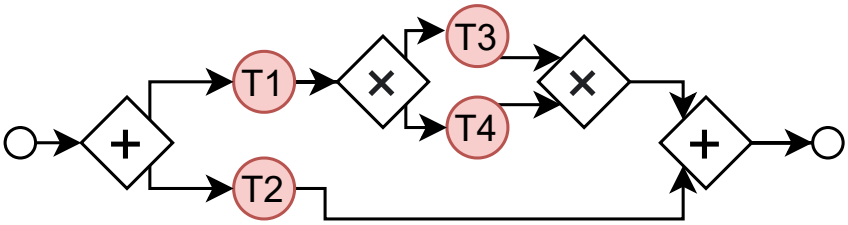


Figure 2.9: Example of a Business Processes Modelling Notation (BPMN) Diagram.

gateways, and *exclusive gateways*, as depicted in Figure 2.10. These *gateways* establish the rules for the flow of activities in the business-process, in the following ways:

- When one of these *gateways* is introduced in an existing branch of the business-process, the branch splits into n other branches, which must all eventually converge into a *gateway* of the same type as the one introduced. For example, the *exclusive gateway* in Figure 2.9 splits into two branches, one of which contains Activity T3 and the other Activity T4. The two branches then converge into another *exclusive gateway*. Similarly, the *parallel gateway* in this Figure also splits into two branches that converge back into a *parallel gateway*.

- A *parallel gateway* functions as an AND, i.e., the activities that belong to the branches leaving the *parallel gateway* must all be executed in order to conclude the execution of the business-process.

- The *inclusive gateway* and *exclusive gateway* both function as an OR, i.e., from the branches leaving these types of *gateways*, only the activities belonging to one of the branches need to be executed in order to conclude the execution of the business-process. The difference between the two is that in the *exclusive gateway*, the branch that is executed is determined by an external condition, whilst in the *inclusive gateway*, the branch that is executed is only known at runtime.

(a) *Parallel gateway*

(b) *Exclusive gateway*

(c) *Inclusive gateway*

Figure 2.10: Business-process *gateways*.

At the second stage (Simulation Stage), BIA performs a simulation of the propagation of a user-chosen threat. This is done by describing the network in MulVAL, which operates using logic programming, i.e., it uses clauses (preconditions and postconditions) and rules. This means that MulVAL receives as input a set of primitives (or preconditions) (rectangle nodes in the attack graph), that define the known information about the network; and a set of rules (AND-nodes/ovals in the attack graph), that define the laws by which the threat propagation is governed. As output, MulVAL produces a set of derivatives (or post-conditions) (OR-nodes/diamonds in the attack graph) that are derived from the application

of the rules to the clauses. For example, defining that a threat T affects a given asset A can be expressed by the primitive $threatExists(A, T)$. The rule "If $compromisedAsset(A)$ and $runsService(A, S)$ then $compromisedService(S)$" can be interpreted as: the derivative $compromisedService(S)$ (i.e., service S has been compromised) is verified if both clauses $compromisedAsset(A)$ and $runsService(A, S)$ are verified (i.e., if the asset A is compromised and runs service S). The output of this stage is an attack graph that contains all the possible propagation paths for the chosen threat.

Finally, the third stage (Impact Assessment) has the goal of examining the impact that the simulated threat has on the organization's infrastructure, by analysing the attack graph. Specifically, it should produce a report that identifies the affected assets, services, activities, and business-processes.

2.3 Summary and Discussion

This chapter has presented different methods for (1) modeling *Cascading Failures* and (2) assessing the impact of cyber-threats. Tables 2.6 and 2.7 summarize each method by categorizing them in different aspects.

Table 2.6 refers to the methods for *Cascading Failures*. Each method is categorized in regard to its context, dynamicity, and failure mode. The context refers to the domain in which the method is applied (either *Critical Infrastructures* or *Supply Chains*). The dynamicity categorizes a method according to its behavior — a dynamic method performs an actual simulation of failures over a network, whereas a static method evaluates the impact of *Cascading Failures* based on a pre-established set of calculations. Lastly, the failure mode reflects how the elements in the system suffer failures — "Complete Fail" means that each of the elements of the systems can only be in one of two states — either operational or failed; on the other hand, "Level of failure/risk", means that instead of being completely operation/failed, each element contains a value (0 to 1) that represents its level of operationality/risk of failure. Regarding the Input, each method either requires a network of corporations/infrastructures containing the dependencies between the various components of the network (Inter-dependencies Network), or it requires a network with the dependencies between different infrastructures (Intra-dependencies Network).

From the methods in Table 2.6, the static methods mostly focus on interdependencies between different *Critical Infrastructures* ([10, 23, 24]). [23] is the one with the more exhaustive evaluation criteria for each infrastructure, which can be a disadvantage since these criteria can be hard to evaluate, or the evaluation can become subjective. By contrast, [24] proposes a similar method, but with simpler evaluation criteria. [10] has the advantage over the previous two of allowing loops in the interdependencies (e.g., infrastructure A depends on B, but B also depends on A).

However, the dynamic methods seem the most promising, since they allow the execution of simulations of failures over the network, which represents a more pragmatic solution, that would likely yield more realistic results regarding *Cascading Failures*. From these, the methods [17–19], in the context of *Supply Chains*, and [16, 25, 26], in the context of *Critical Infrastructures*, are very similar — all of them propose simulations based on interdependencies between the graph's nodes and failures due to overload/underload of said nodes. The main difference between them is the use of *betweenness* for

Table 2.6: Summary of *Cascading Failures* methods.

		[17, 18]	[19]	[23]	[24]	[10]	[16, 25], [26]	[27]	[28]
Context	CI			✓	✓	✓	✓	✓	✓
	Supply Chain	✓	✓						
Dynamicity	Dynamic	✓	✓				✓	✓	
	Static			✓	✓	✓			✓
Failure mode	Level of Failure/ Risk			✓	✓	✓			✓
	Complete Failure	✓	✓				✓	✓	
Input	Inter-dependencies Network	✓	✓				✓	✓	✓
	Intra-dependencies Network			✓	✓	✓			
	Other	✓[1]	✓[1,2]	✓[3]	✓[4]	✓[5]	✓[6]	✓[7]	✓[7]

[1] Strengths of edges and loads (λ, δ); Upper and lower bounds (α, β)
[2] Overload parameter (γ)
[3] Set of criteria for each sub-sector (C_T, C_{EV}, C_{EM})
[4] Frequency of initiating event (f); Probability (p), Extent (e) and Duration (d) of failure in each infrastructure
[5] Level of dependency between each pair of sub-sectors (a_{ki}); Internal risk of sub-sector (c_k)
[6] Tolerance parameter (α)
[7] Probability of failure (p), Recovery time (t), influence on downstream functions (f) for each function

the calculation of the nodes' capacity and initial load ([16, 19, 25, 26]), as opposed to simply using the number of neighbours of each node ([17, 18]). The simulation method offered by [27] is slightly different, since it is time-bound and allows the nodes to recover after a failure. This means, though, that the excepted downtime for each node will have to be inputted, but it has the advantage of being able to estimate, for example, the average amount of failures and total duration of failures for each node during the simulation time.

Looking at the "Input" line of Table 2.6, it is clear that most of the methods used to predict *Cascading Failures* require some sort of metric capable of quantifying the level of damage inflicted by the event that initiates the *Cascading Failures* (e.g., set of failing nodes/sub-sectors, frequency/probability/extend/duration of the failure). Hence, in order to predict the *Cascading Failures* that can result from the exploitation of cyber-threats, it can be concluded that these methods would benefit from a methodology capable of quantifying the impact inflicted by cyber-attacks on the attacked organization/infrastructure, which would serve as a starting point to simulate the extent of the propagation of *Cascading Failures* among other organizations/infrastructures dependent on the attacked organization/infrastructure.

Table 2.7 categorizes each of the discussed impact assessment methods in regard to their type of propagation model, randomness, dynamicity, input metrics, and assessment layers. The type of propagation model refers to the type of tool used by the method to model impact propagation (either Risk Assessment Graph, Vulnerability-Asset-Service-Mission model, logic-based attack graph, or Bayesian-based attack graph). The randomness refers to the probabilistic nature of the method — if the method

always produces the same output for a given input, it is deterministic, on the other hand, if it contains random variables that follow a probability distribution, it is stochastic. The dynamicity indicates whether a method can react to changes in the network characteristics — a dynamic method is able to propagate changes, while a static method needs to start from the beginning whenever there is a change in the network. Input metrics refer to the input parameters required by the method, where CVSS refers to the CVSS scores of vulnerabilities. Finally, the Assessment Layers correspond to the abstraction layers that are considered in the model (Vulnerability, Asset, Service, and Mission).

Table 2.7: Summary of cyber-threats impact assessment methods.

			[45]	[46]	[42]	[41]	[33]	[47]	[58]	[48], [49], [50]	[12], [11]	BusICalc
Type of propagation model	RAG		✓									
	VASM			✓	✓						✓	✓
	Attack Graph	Logic				✓	✓	✓	✓		✓	✓
		Bayesian								✓		
Randomness	Deterministic		✓	✓	✓	✓	✓		✓	✓	✓	✓
	Stochastic							✓				
Dynamicity	Dynamic		✓	✓	✓		✓	✓	✓	✓	✓	✓
	Static					✓						
Input Metrics	CVSS		✓	✓	✓	✓	✓	✓		✓		✓
	Other		✓[1]	✓[2]		✓[3]			✓[4]	✓[5]	✓[6]	✓[6]
Assessment Layers	Vulnerability		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Asset		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Service			✓	✓	✓	✓	✓	✓	✓	✓	✓
	Mission			✓	✓						✓	✓

[1] Exploitability growth rate (α_w)
[2] Vulnerabilities study level; Level of defense measures; Attack timeliness
[3] Attack strategy preference; "Out-of-bounds" attack success probability
[4] Existing countermeasures; Intrusion history; Password policy
[5] Exploitability propagation probabilities
[6] Mappings between: Assets and Type of Asset; Type of Asset and Vulnerabilities; Assets and Business-Processes

Regarding the vulnerability assessment, from the two methods analysed, DREAD ([54]) and CVSS ([40]), CVSS has the advantages of being widely used, publicly available, and being less subjective.

As for the methods that study the impact propagation of cyber-threats, the use of a Risk Assessment Graph ([45]) presents a complete model of the IT network, but the method is limited to the impact at the asset level. The VASM model ([42, 46]) solves this problem by modelling the service and mission layers as well. While the model in [46] is quite complex and requires metrics about the vulnerabilities that can be subjective, the propagation model in [42] is simpler and yields similar results. Attack graphs ([33, 41, 47–50, 58]) represent a very efficient way of analysing impact propagation in IT networks, since they already contain all the possible attack paths. From these, the method [41] is able to determine the level of criticality of each of the nodes based on quite exhaustive metrics with minimal subjectivity.

[33] proposes the simpler propagation model from all the methods, that is also used by [47], though this last one employs the Monte Carlo stochastic method in order to obtain a probability distribution for the impact. Lastly, BIA ([11, 12]) proposes a hybrid method that uses a version of the VASM method to model the entities of the system, and attack graphs to simulate the propagation of threats.

This work will take advantage of both BIA ([11, 12]) and the method proposed by Jakobson [42] in order to construct a methodology capable of simultaneously simulating the propagation of a chosen cyber-threat through an organization, and estimating a metric for the impact of this propagation. The last column of Table 2.7 shows how the proposed methodology — BusICalc — compares to the remaining methods studied in the related work. Since it corresponds to an extension of the BIA methodology, it uses the same type of propagation models (VASM and logic-based attack graphs), and models the same assessment layers - Vulnerability, Asset, Service, and Mission. The main difference is thus in the ”Input Metrics” row, since in order to calculate a metric for the impact of a cyber-threat propagation, it is necessary an initial assessment of the severity of the threats on a given asset — for this purpose the CVSS score is used.

Since it is just a model, BusICalc inevitably lacks some of the realism of the actual system being modeled, which is a result of the assumptions that are made in order to obtain a method that is relatively simple to use. Some of these assumptions include:

- The CVSS Base Score is used to quantify the severity of a threat. Here, two simplifications are made — first, it is assumed that a threat exploits a specific vulnerability on a given asset, and the CVSS score of the vulnerability is used to determine the severity of the threat; and second, it is assumed that the impact inflicted on the attacked asset is directly proportional to this severity. Although not ideal, the CVSS score is widely used in methods that study the propagation of impact of cyber-attacks in IT networks ([33, 41–50]), since it provides a metric that would otherwise require expert knowledge to be estimated.

- The methodology used to calculate the propagation of impact, based on the method proposed by Jakobson [42], also makes a few assumptions, namely that if a service that depends upon a single asset is compromised, then the impact of the service will be the same as the impact of the asset it runs on. In reality, this behaviour might not be the best, since the level of compromise of a service might depend on other factors rather that solely on the level of compromise of the affected asset.

- Lastly, the threat propagation method that originates from BIA ([11, 12]) assumes that any type of threat can be leveraged by an attacker in order to move laterally across a network by exploiting other assets, whereas realistically not all types of threats would grant the attacker access to the neighbouring assets.

Chapter 3

Solution

The proposed approach — Business Impact Calculator (BusICalc) — was designed with the goal of simulating the propagation of a user-chosen cyber-threat throughout an organization, and estimating the impact of that propagation on the organization's business-processes, by yielding an impact metric.

As a result, BusICalc improves upon the *Business Impact Assessment* methodology ([11, 12]). The main weakness of BIA is not providing a quantitative metric for the impact of a given attack. BusICalc aims at enhancing BIA by computing an estimate of the impact of a BIA simulation.

In this chapter, Section 3.1 describes the problems that BusICalc proposes to solve; Section 3.2 presents how the propagation paths are handled and the various ways they can be merged together; Section 3.3 presents the algorithm for impact calculation over the propagation paths; and lastly Section 3.4 presents a summary of the chapter.

3.1 Problem Description

As already mentioned, the main goal of this approach is improving upon BIA by providing a methodology for quantifying the impact of a threat to a business organization. In this context, it is important to define the concept of *impact*: *impact* refers to the loss of operationality of an organization's business-processes, considering that a specific attack has occurred, characterized by a specific entry-point threat and propagation path through the organization's network. This concept differs from the *probability* that an organization will be attacked, as well as from the *risk* that an organization is under (in fact, the *risk* is often given by the product of *impact* and *probability* [60]).

Based on this definition, and considering that BIA uses a version of the VASM model to represent the network, the most adequate method for impact estimation, present in the literature, is the method proposed by Jakobson [42]. Hence, BusICalc will adapt this method to BIA for the purpose of impact calculation, as described in Section 3.3. Since there are some variations between the different entities belonging to the two methods, some adjustments need to be made, namely:

- In Jakobson [42] there are direct dependencies between computers, while in BIA a propagation path from one computer to the next goes through subnets and routers. Besides, in Jakobson [42],

the concept of entry-point asset/threat is not used. Instead, it is assumed that all existing threats are exploited simultaneously. In order to adapt Jakobson [42], it will be assumed that an asset has a dependency on another if BIA determines that there is a direct propagation path (eventually passing through subnets and routers) between the two assets. Hence, the threats that will be considered will be solely the ones that affect the compromised assets determined by BIA, after the selection of the entry-point.

- Jakobson [42] contains AND and OR dependencies between the nodes assets and services, and between services and activities. A node can AND-depend on two other nodes, meaning the two dependency nodes are necessary for the dependent node to be provided, or it can OR-depend on the two nodes, meaning only at least one of the two dependency nodes is necessary for the dependent node to be provided. In BIA, however, these dependencies do not exist. Hence, as explained in Section 2.2.8, it is assumed that any relationship of this type is an OR-dependency, meaning, for example, that if a service depends on several assets, it is assumed that it is an OR-dependency.

- In Jakobson [42], the flow of activities in the business-processes (referred to as ”missions” in [42]) is not governed by *inclusive/exclusive/parallel gateways*, as in BIA, but instead by AND/OR nodes. This, however, does not constitute a problem, since the AND nodes act similarly to *parallel gateways*, and OR nodes act similarly to *inclusive*/*exclusive gateways*.

With these adjustments, the method employed for impact computation is presented in Section 3.3.

The second limitation with BIA that BusICalc tries to overcome is related to the fact that the propagation paths discovered by BIA are very simple paths that, individually, do not make the simulation of complex attacks possible. The solution for this problem is to merge several of these individual paths, as demonstrated in Section 3.2.

3.2 Propagation Paths

As mentioned in Section 2.2.8, BIA uses a set of primitives to translate the network model into MulVAL, so that MulVAL is able to output the propagation paths, given an entry-point provided by the user. The propagation paths that are identified by MulVAL can be referred to as *trivial paths*. They receive this designation because they only contemplate a single route from a threat to an activity belonging to a business-process. In practice, it means that each of these paths will start at a user-selected threat, then it will contain a series of assets through which the threat propagates, and finally a single service and a single activity belonging to a business-process.

In order to discard paths that contain infinite loops, it is also assumed that these trivial paths cannot go through the same node twice, which is particularly relevant for Router and Subnet nodes. It means that, for instance, for the example of Figure 2.8, the path $A3 \rightarrow Subnet2 \rightarrow A4 \rightarrow Subnet2 \rightarrow Router1 \rightarrow Subnet1 \rightarrow A1$ will not be considered, since it contains the $Subnet2$ twice. This restriction does not interfere with the compromised nodes, i.e., the same nodes (Assets, Services, Activities) are compromised

regardless of this restriction, if the whole set of trivial paths is considered. It does however significantly reduce the number of trivial paths that are computed since only the trivial paths with two or fewer assets are considered.

Figure 3.1 presents the four trivial paths for the network of Figure 2.8 in which the entry-point is a threat on Asset A3.

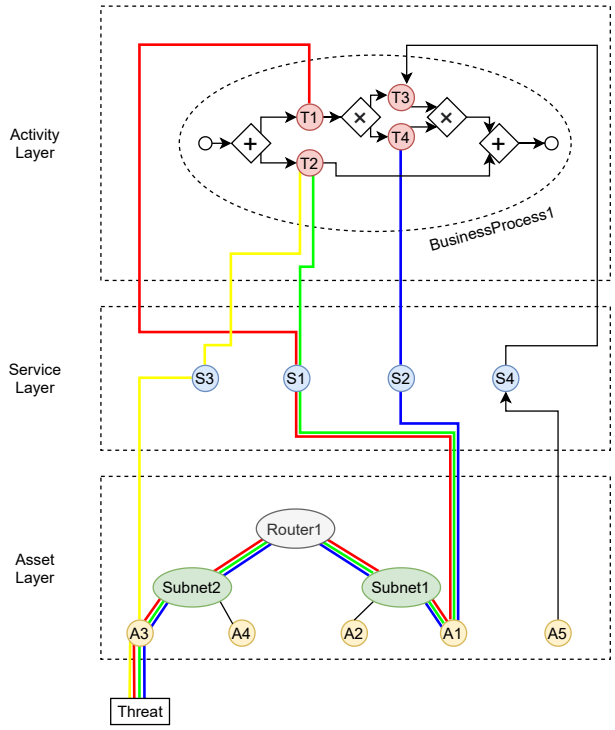


Figure 3.1: Four trivial paths (yellow, red, green, blue) for the entry-point in Asset A3.

The main limitation with considering only the trivial paths individually is that it becomes impossible to obtain a single value for the impact of a complex attack. For example, suppose an attacker gains control over a given asset and decides to compromise all the services that are run by that asset. In this scenario, for each compromised service there would be at least one trivial path, since each trivial path only contains one service. This implies that the only way to study the impact of the attack would be to apply the impact quantification algorithm to each of these n trivial paths that correspond to the attack. The result would be n different values of impact, which would be hard to interpret.

This work proposes merging all the trivial paths that result from a given attack scenario into a single merged path. This would allow the algorithm in Section 3.3 to be applied to a single path, and as a result, only one value for the impact would be obtained, making it easier to analyse. For instance, suppose an attack scenario in which the attacker is able to simultaneously disable access to the organization's database, and shut down the server that handles the organization's web application. In this model, this scenario would correspond to the simultaneous compromise of two different activities (e.g., *access database* and *visit web application*), which would necessarily encompass at least two different trivial paths (since each trivial path can only contain one activity). As a result, without the option of merging trivial paths, there is no possibility of estimating the impact of such a scenario, since it would only be possible to study the impact of each activity being compromised separately. However, by allowing trivial

paths to be merged, it becomes possible to estimate the impact of complex attacks such as the one described, and as a result obtain an impact metric that considers the simultaneous compromise of the two activities.

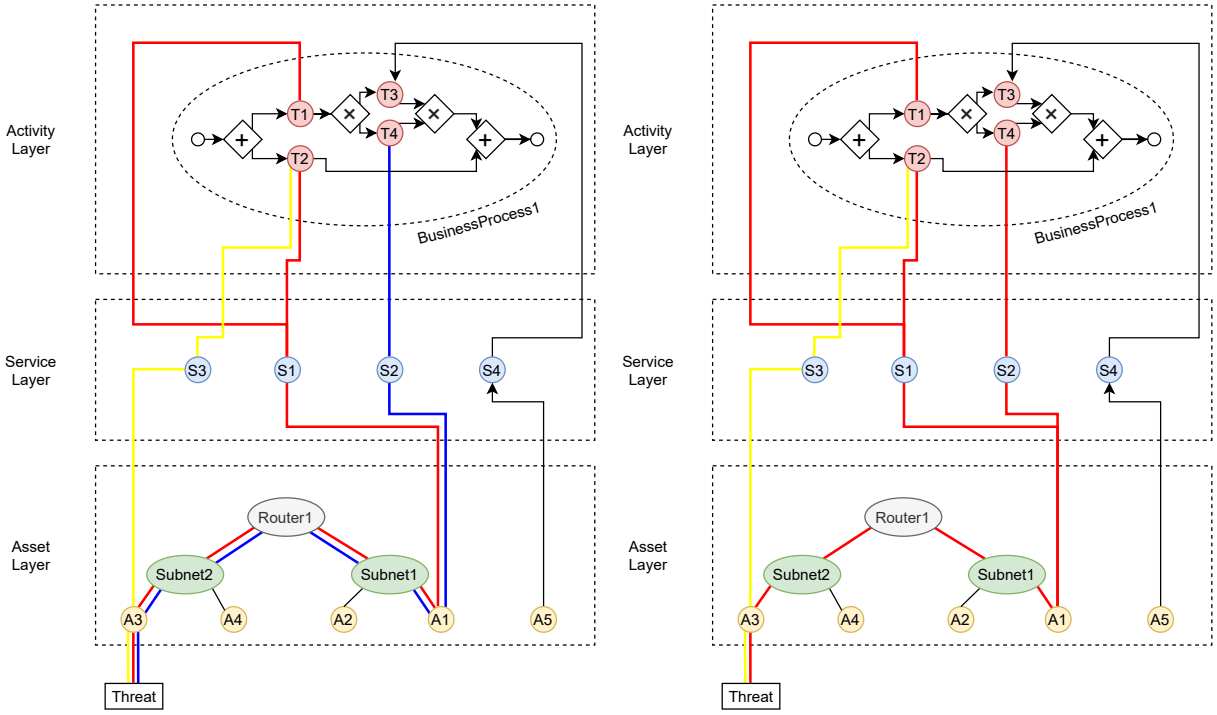
For this reason, BusICalc gives the user the liberty to merge the trivial paths in one of the following ways:

- *No Merge* — The paths considered are the trivial paths provided by BIA.
- *Merge Activities* — Assumes that when the attacker compromises a given service, he/she also compromises all the activities that are provided by it, i.e., merge the trivial paths that are equal up to service. This mode can be used in situations where the services are tightly-bound to the activities they provide, in such a way that the compromise of the service automatically causes its activities to be compromised.
- *Merge Services* — Assumes that when the attacker compromises a given asset, he/she also compromises all the services that run in it, i.e., merge the trivial paths that are equal up to the last asset. This mode is useful if the services are considered to be automatically affected if the asset they run on is affected.
- *Merge Assets* — Assumes that the attacker is able to compromise all the trivial paths that result from a given entry-point threat, i.e., merge all of the trivial paths. This mode is useful to simulate the worst-case scenario of an attack that exploits a given entry-point.
- *Custom Merge* — Merges a set of user-selected trivial paths. This mode can be used either to manually select a set of trivial paths to merge, or to merge trivial paths according to some condition not contemplated in the previous modes. For example, it can be useful to group all the trivial paths that affect a specific business-process.

Figure 3.2 presents, for the example network, the paths that result from the options *Merge Activities*, *Merge Services*, and *Merge Assets*.

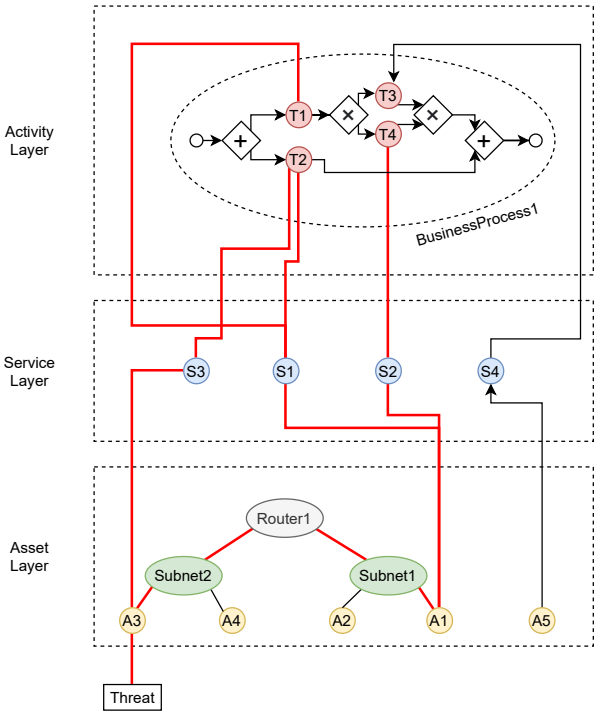
3.3 Impact Calculation

This section will describe the algorithm developed for impact calculation in BusICalc. The goal of the algorithm is to compute a value for the impact of the propagation of the entry-point threat (chosen by the user), which is propagated through a given path $\mathcal{P}$ — $I_{\mathcal{P}}$ — (also referred to simply as the impact of the path $\mathcal{P}$), computed by Equation 3.8. The purpose of this metric is estimating the impact that a given cyber-threat may cause, once exploited, to the business-processes of the organization. Hence, this metric can assume any value between 0 and 1. If a given path $\mathcal{P}$ has an impact of 0 (i.e., $I_{\mathcal{P}} = 0$), then the business-processes were unaffected by the propagation of the entry-point threat through the path $\mathcal{P}$. On the other hand, if the path has an impact of 1 (i.e., $I_{\mathcal{P}} = 1$), then the propagation of the entry-point threat through the organization using path $\mathcal{P}$ has resulted in the complete lack of operationality of the business-processes of the organization.



(a) Merge Activities

(b) Merge Services



(c) Merge Assets

Figure 3.2: Paths that result from the merge of trivial paths according to the merge options — *Merge Activities*, *Merge Services* and *Merge Assets* — for the entry-point in Asset A3.

Considering that:

- T is the set of all threats;
- D is the set of all assets (devices);

- S is the set of all services;
- A is the set of all activities;
- BP is the set of all business-processes;
- $T(d) \subset T$ is the set of threats that affect asset $d \in D$;
- $D(s) \subset D$ is the set of assets that run service $s \in S$;
- $S(a) \subset S$ is the set of services that provide activity $a \in A$;
- $A(bp) \subset A$ is the set of activities that support the business-process $bp \in BP$;
- $E(bp)$ is the set of execution threads belonging to the business-process $bp \in BP$;
- $d_{entrypoint} \in D$ is the entry-point asset (user-chosen);
- $t_{entrypoint} \in T$ is the entry-point threat under analysis (user-chosen);
- $P_i = (D^i, s^i, a^i)$ defines a trivial path, in which:
  - $D^i = \{d_0^i, \ldots, d_K^i\} \subset D$ is the set of affected assets, ordered such that d_0^i is the entry-point asset ($d_0^i = d_{entrypoint}$), d_1^i is the next asset compromised, and so on;
  - $s^i \in S$ is the affected service;
  - $a^i \in A$ is affected activity;
- $\mathcal{P} = \{P_1, \ldots, P_N\}$ represents a generic merged path that aggregates the trivial paths $P_1, \ldots, P_N$. Here, it is a necessary condition that $d_0^1 = d_0^2 = \cdots = d_{entrypoint}$, i.e., the entry-point asset is the same for all trivial paths that comprise the merged path;
  - $\mathcal{S} = \{s^i | \forall P_i \in \mathcal{P}\}$ is the set of services affected by the merged path $\mathcal{P}$ (which corresponds to the set of services affected by each of the trivial paths that comprise $\mathcal{P}$);
  - $\mathcal{A} = \{a^i | \forall P_i \in \mathcal{P}\}$ is the set of activities affected by the merged path $\mathcal{P}$ (which corresponds to the set of activities affected by each of the trivial paths that comprise $\mathcal{P}$);
- IF_t is the Impact Factor of threat t;
- OC_d^{Asset}, $OC_s^{Service}$, $OC_a^{Activity}$, $OC_e^{ExecutionThread}$, $OC_{bp}^{BusinessProcess}$ are, respectively, the Operational Capacities of asset d, service s, activity a, execution thread e, and business-process bp;
- $I_{\mathcal{P}}$ is the impact of the propagation of the entry-point threat through the path $\mathcal{P}$.

The algorithm starts by calculating the Impact Factor (*IF*) of each threat. Here, the rationale is that a threat exploits a specific vulnerability in an asset. So, the Impact Factor of the threat, which measures the degree to which it is capable of compromising the attacked asset, is calculated based on the CVSS

score of the compromised vulnerability, according to Equation 3.1, in order to obtain a value between 0 and 1 (since the CVSS score ranges between 0 and 10).

$$IF_t := \frac{CVSS_t}{10}, \forall t \in T \tag{3.1}$$

The algorithm then assigns to each node — Asset, Service and Activity — an Operational Capacity (*OC*) of 1 (Equation 3.2). This parameter is a measure of the operationality of the node, that can assume values between 0 and 1, where a value of 1 means the node is fully operational and a value of 0 means the node is completely inoperable. By assigning a value of 1 in the beginning, the assumption is that every node starts fully operational before the simulated attack.

$$\begin{cases} OC_d^{Asset} := 1, \forall d \in D \\ OC_s^{Service} := 1, \forall s \in S \\ OC_a^{Activity} := 1, \forall a \in A \end{cases} \tag{3.2}$$

Then, for each trivial path that composes the generic path $\mathcal{P}$, the algorithm will update the assets' OCs according to Equation 3.3 — the Operational Capacity of the asset directly affected by the entry-point threat (i.e., the entry-point asset) is decreased by an amount equal to the Impact Factor of the entry-point threat, whereas the OCs of the remaining assets in the trivial path are either updated to the Operational Capacity of the previous asset in the path, or are lowered by an amount equal to the Impact Factor of their most impactful threat, depending on whichever yields a smaller value. This means that the OC of the previous asset is carried over directly to the next asset, unless the next asset is affected by some threat that would make this value lower, in which case it is assumed that the attacker is able to compromise this threat and lower the OC of the asset.

$$\forall P_i \in \mathcal{P} : \begin{cases} OC_{d_0^i}^{Asset} := max(1 - IF_{t_{entrypoint}}, 0) \\ OC_{d_n^i}^{Asset} := min(OC_{d_{n-1}^i}^{Asset}, max(min_{t \in T(a_n^i)}(1 - IF_t), 0)), \quad n = 1, \ldots, K \end{cases} \tag{3.3}$$

Next, the algorithm will update the Operational Capacities of the services affected by the path $\mathcal{P}$ — $s \in \mathcal{S}$ — according to Equation 3.4. In practice, it means that the OC of each affected service will be updated to the average of the OCs of the assets that run that service. The reason why the avg operator is used is because it is assumed that each asset runs the full service, as explained in Section 3.1, which corresponds to an OR-node in the method described in Jakobson [42]. If, instead, every asset was necessary to run the service, this would correspond to an AND-node, and the operator min would be used instead of avg.

$$OC_s^{Service} := avg_{d \in D(s)}(OC_d^{Asset}), \forall s \in \mathcal{S} \tag{3.4}$$

Likewise, for the affected activities — $a \in \mathcal{A}$ — their Operational Capacities are updated according to Equation 3.5, to the average of the OCs of the services that provide each activity.

$$OC_a^{Activity} := avg_{s \in S(a)}(OC_s^{Service}), \forall a \in \mathcal{A} \tag{3.5}$$

The next step is computing the OCs of the business-processes. In order to understand how they are computed, it is first necessary to understand the concept of *execution threads*. An execution thread corresponds to a minimum sequence of activities that, once executed, concludes the execution of the business-process. For example, consider the business-process bp depicted in Figure 2.9. Since it contains an *exclusive gateway*, only one of either activity $T3$ or $T4$ needs to be executed in a given execution instance. For this reason, the business-process has the following execution threads:

- $e_1 = \{T1, T2, T3\}$;
- $e_2 = \{T1, T2, T4\}$.

In this case, although the set $\{T1, T2, T3, T4\}$ would also conclude the execution of the business-process, it is not considered an execution thread, since it is not a "minimum sequence", i.e., it contains redundant activities (either $T4$ or $T3$ could be removed). Hence, the set of execution threads of business-process bp is solely comprised of e_1 and e_2, i.e., $E(bp) = \{e_1, e_2\}$.

With this, the algorithm will compute, for each business-process, the OCs of all its execution threads, according to Equation 3.6, i.e., the OC of an execution thread is the product of the OCs the activities that comprise it.

$$\begin{aligned} &\forall bp \in BP : \\ &\quad OC_e^{ExecutionThread} := \prod_{a \in e}(OC_a^{Activity}), \forall e \in E(bp) \end{aligned} \tag{3.6}$$

The last Operational Capacities computed are the OCs of business-processes. The OC of a business-process is computed by averaging the OCs of the execution threads that belong to it, as defined in Equation 3.7.

$$OC_{bp}^{BusinessProcess} := avg_{e \in E(bp)}(OC_e^{ExecutionThread}), \forall bp \in BP \tag{3.7}$$

Finally, the impact of the generic path $\mathcal{P}$ on the organization — $I_{\mathcal{P}}$ — (i.e., the impact of the propagation of the entry-point threat through path $\mathcal{P}$), is given by the average of the loss of operationality of the business-processes that belong to the organization, as in Equation 3.8, where N_{BP} corresponds to the total number of business-processes. This Equation yields a value between 0 and 1 for the impact.

$$I_{\mathcal{P}} = \frac{\sum_{bp \in BP}(1 - OC_{bp}^{BusinessProcess})}{N_{BP}} \tag{3.8}$$

3.4 Summary

This chapter has presented the goal of BusICalc and the problems it tries to solve. Namely, it has demonstrated how impact calculation can be applied to BIA in order to build a methodology to determine

a quantitative value for the impact of a cyber-attack.

The method used for impact estimation, detailed in Section 3.3, is based on Jakobson [42]. This method is adequate to work with BIA, since both use a similar layered model to describe the network elements (Assets, Services, Activities, Business-Processes) and the various dependencies between the different elements. There are, though, some modifications that need to be made, due to the fact that some minor differences exist between the two models. These adjustments are explained in Section 3.1. It is also described how BusICalc is able to model complex attacks by merging the trivial paths that result from the BIA simulations (Section 3.2).

Chapter 4

Implementation

This chapter presents how the proposed solution in Chapter 3 (BusICalc) is implemented into a single software tool that will enable to test its practical viability. This chapter is divided into three sections — the first of which (Section 4.1) presents an overview of the architecture of BusICalc; the remaining sections each explains the implementation of one of the modules that comprise the application — the Setup Module (Section 4.2) and the Impact Calculation Module (Section 4.3).

Due to its resourcefulness, *Python*[1] programming language was used for the development of these modules.

4.1 System Architecture

The architecture of BusICalc is illustrated in Figure 4.1. The system is composed of two main modules. In the first module — *Setup Module* — BIA is invoked in order to obtain the propagation paths for a business network given an entry-point. Then in the second module — *Impact Calculation Module* — the impact is calculated for a set of paths that are generated according to user input.

The user must first provide in the file *setup_config.json* the address of the server in which BIA is running with the scenario meant to be evaluated, as well as the address of the Neo4j Database used by BIA, and the corresponding login credentials. The last parameter needed from the user is the entry-point, i.e., the pair (asset, threat) that will be used by BIA to simulate the attack and produce the propagation paths starting in said entry-point.

In the Setup phase, the sub-module *setup.py* will start by requesting a simulation from BIA with the entry-point selected by the user. BIA will then output the file *bia_output.json*. This file contains information about the network links that were exploited during the simulated attack. The sub-module *populate_network.py* will use this file, in conjunction with the Neo4j Database, where the whole structure of the network is stored, to build the *network* object, which will contain the assets, services, activities, and business-processes, as well as all the relations between all these entities. The last sub-module of the setup phase — *compute_paths.py* — will identify, from the *network* object produced by the previous

[1] https://www.python.org/

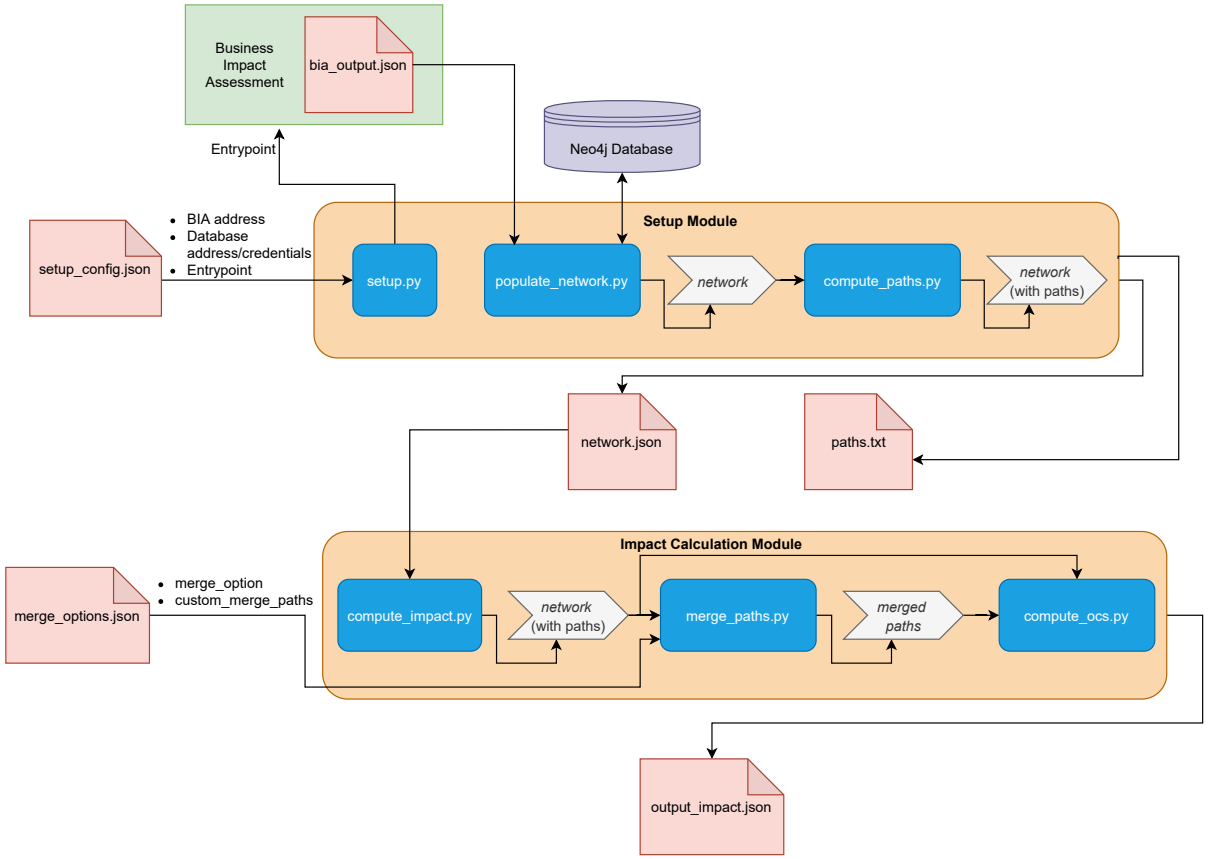


Figure 4.1: Architecture of BusICalc.

sub-module, the trivial paths that were produced by BIA starting at the selected entry-point. The output of the setup module will be a *.json* file containing a representation of the network object, as well as a user-readable file — *paths.txt* — containing the trivial paths identified.

The user will then have the liberty to choose the paths on which the impact of the threat on the network will be calculated. This choice is made by configuring the file *merge_options.json*. This configuration will then be used to merge the trivial paths according to user specification. The second module — *Impact Calculation Module* — will start by importing the *network.json* file generated by the previous module in order to reconstruct the *network* object (this is done by the sub-module *compute_impact.py*), and then the sub-module *merge_paths.py* will merge the network's trivial paths according to the option specified by the user. For each of the paths produced by this sub-module, the next sub-module — *compute_ocs.py* — will apply the algorithm described in Section 3.3 in order to determine the impact of each of them on the business network. The impact of each path will be presented in the file *output_impact.json*.

4.2 Setup Module

In order to execute the attack simulation in BIA, it needs to be running with the scenario meant for evaluation on a server whose address is the one specified in the field *BIA_address* of the file *setup_config.json*. Then, the simulation is solicited by making an HTTP GET request to the URL

`http://[BIA_address]/simulate` with the parameters *entry-point* equal to the IP address of the entry-point asset and *threat* equal to the name of the entry-point threat. By using *python*'s *requests* library, this operation is performed with a single line, as shown in Listing 4.1.

Listing 4.1: HTTP request to the BIA application.

```
1 requests.get("http://" + BIA_address + "/simulate?entrypoint=" + entrypoint_asset_ip + "&
    threat=" + entrypoint_threat_name)
```

Once the simulation is performed, the results can be requested by sending an HTTP GET request to the URL `http://[BIA_address]/export-results`. The result of this request will be a *.json* file (*bia_output.json*) containing the compromised nodes (Assets, Services, Activities) and links, that have resulted from the simulation.

This file, however, does not contain all the information necessary for BusICalc. Hence, direct access to the database used by BIA is also necessary. BIA uses the Neo4j database Platform[2]. In order to get access to this database, the `neo4j.GraphDatabase` library from *Python* is used to initialize the database's *driver* with the database server address, username, and password (all configured in the file *setup_config.json*). This *driver* will then allow queries to be made to the database directly.

The sub-module *populate_network.py* will use both the *bia_output.json* file and queries to the database in order to build a *network* object. This object will contain information regarding all Business-Processes, Activities, Services, Assets, Threats, as well as all the relationships between these entities (acquired from the database). Besides, the *network* object will also contain the *Subnets* and *Routers* through which the threat was propagated, according to BIA's simulation, which are taken from the *bia_output.json* file.

The next sub-module — *compute_paths.py* — will compute, from the *network* object, the trivial paths of this network, starting at the entry-point asset. For this purpose, a depth-first search algorithm is employed, whose pseudo-code is presented in Listing 4.2. In this function, the input parameters are: $start\_node$ — the asset/router/subnet from which the discovery of paths will start; $current\_path$ — a list of assets/routers/subnets belonging to the path currently being discovered; $network$ — the $network$ object to which the discovered paths will be added.

The algorithm starts by checking whether the $start\_node$ already belongs to the $current\_path$. Since a trivial path is not supposed to have duplicate nodes, the node is only added to the path in case the path does not yet contain it (lines 2-4).

Then, the discovery process continues by recursively calling this function on the next node in the path (lines 5-17). According the type of $current\_node$ (asset, subnet or router), the next node in the path can be: a subnet if the $current\_node$ is an asset (lines 5-7); a router or an asset if the $current\_node$ is a subnet (lines 8-12); a router or subnet if the $current\_node$ is a router (lines 13-17). Since in *Python* the objects are passed by reference, it is necessary to create a copy of the $current\_path$ before invoking the function $explore\_paths\_starting\_in\_node$ (lines 7, 10, 12, 15, 17). By creating this new object with each invocation, whenever there is a split in the current path (e.g., from a router are reachable two distinct subnets), each of the created copies will correspond to one of the diverging paths.

[2]https://neo4j.com/

In case the $current\_node$ is an asset, the algorithm will explore in the service, activity, and business-process layers, the nodes that can be reached from that asset (lines 19-25). Again, by creating a copy of the path for every distinct set of (service, activity, business-process) (line 22), it is guaranteed that each discovered trivial path will only contain one service, one activity, and one business-process.

This trivial path is finally added to the $network$ object (line 26).

Listing 4.2: Algorithm for discovering trivial paths.

```
1  def explore_paths_starting_in_node(start_node, current_path, network):
2      if start_node in current_path:
3          return #make sure a node does not appear twice in the trivial path
4      current_path.addNode(curent_node)
5      if isAsset(start_node):
6          for subnet in start_node.getSubnets():
7              explore_paths_starting_in_node(subnet, current_path.copy(), network)
8      elif isSubnet(start_node):
9          for router in start_node.getConnectedRouters():
10             explore_paths_starting_in_node(router, current_path.copy(), network)
11         for asset in start_node.getConnectedAssets():
12             explore_paths_starting_in_node(asset, current_path.copy(), network)
13     elif isRouter(start_node):
14         for router in start_node.getConnectedRouters():
15             explore_paths_starting_in_node(router, current_path.copy(), network)
16         for subnet in start_node.getConnectedSubnets():
17             explore_paths_starting_in_node(subnet, current_path.copy(), network)
18     if isAsset(start_node):
19         for service in start_node.getServices():
20             for activity in service.getActivities():
21                 for business_process in activity.getBusinessProcesses():
22                     complete_current_path = current_path.copy()
23                     complete_current_path.addService(service)
24                     complete_current_path.addActivity(activity)
25                     complete_current_path.addBusinessProcess(business_process)
26                     network.addTrivialPath(complete_current_path)
```

By initializing this algorithm with the entry-point asset in parameter *start_node*, the function will recursively discover all trivial paths and add them to the *network* object.

This object is then stored in the *network.json* file to be used by the next module.

4.3 Impact Calculation Module

The first task of the *Impact Calculation Module* will be reconstructing the *network* object from the *network.json* file containing its representation. Then, according to the user-specified configuration parameters regarding the merging of paths (in the *merge_options.json* file), the sub-module *merge_paths.py* will produce the merged paths, over which the impact will ultimately be calculated by the sub-module *compute_ocs.py*, according to the algorithm in Listing 4.3.

This algorithm will compute the impact for each of the merged paths individually. For this reason, before starting the calculation of Operational Capacities, it initializes the Operational Capacities (OCs) of all nodes (assets, services, activities, business-processes) to their default value — 1 (line 3), and also retrieves the Impact Factor (IF) of the entry-point threat (line 4).

Then, for each of the trivial paths that compose the merged path, it will update the Operational Capacities of the assets that compose each trivial path, according to the Equation 3.3, explained in Section 3.3 (lines 5-15). For each of the services, activities, and business-processes affected by the merged path, their Operational Capacities will be updated according, respectively, to Equations 3.4, 3.5 and 3.7 (lines 16-23).

Lastly, the impact of the merged path is computed according to Equation 3.8 (line 24).

Listing 4.3: Algorithm for determining Operational Capacities and Impact.

```
1  def compute_OCs(merged_paths, network):
2      for path in merged_paths:
3          initializeOCs() #restore OCs of all nodes to 1
4          IF_entrypoint := path.getEntrypointThreat().getImpactFactor()
5          for trivial_path in path:
6              first_asset := trivial_path.firstAsset()
7              first_asset.OC := min(first_asset.OC, max(1-IF_entrypoint, 0))
8              prev_asset := first_asset
9              current_asset := trivial_path.getNextAsset(first_asset)
10             while current_asset is not null:
11                 current_asset.OC := min(current_asset.OC, prev_asset.OC)
12                 for threat in current_asset.getThreats():
13                     current_asset.OC := min(current_asset.OC, 1-threat.getImpactFactor())
14                 prev_asset := current_asset
15                 current_asset := trivial_path.getNextAsset(current_asset)
16         for service in path.getAffectedServices():
17             service.OC := average over [asset.OC for asset in service.getAssets()]
18         for activity in path.getAffectedActivities():
19             activity.OC := average over [service.OC for service in activity.getServices()
       ]
20         for process in path.getAffectedProcesses():
21             for execution_thread in process.getExecutionThreads():
22                 execution_thread.OC := product over [activity.OC for activity in
       execution_thread.getActivities()]
23             process.OC := average over [execution_thread.OC for execution_thread in
       process.getExecutionThreads()]
24        path.impact :=  average over [(1-process.OC) for process in path.getProcesses()]
```

4.4 Summary

This chapter has presented how BusICalc was implemented into an actual tool, and explains the operating principles behind the developed algorithms.

In particular, it is demonstrated how the communication between the Setup Module and BIA is achieved, and how this module is able to directly query BIA's database, in order to establish a representation of the network being studied. Moreover, an explanation is given of the algorithm used by the Setup Module to compute all the trivial propagation paths.

Regarding the second module — Impact Calculation Module — this chapter details how the trivial paths are merged and describes how the algorithm for impact calculation, firstly introduced in Section 3.3, is accomplished.

The implementation of BusICalc makes use primarily of *Python* programming language, as well as of typical input/output file types (*JSON* and *TXT*).

Chapter 5

Evaluation

This chapter aims to test BusICalc in a realistic scenario in order to verify its viability. Section 5.1 presents the details of the testbed used for this purpose. In Section 5.2 are showcased the scenarios used to test BusICalc in the presented testbed, and the results are analysed. Finally, Section 5.3 presents a summary of the evaluation.

5.1 Evaluation Setup

The dataset used to evaluate BusICalc is based on the EPIC (Electric Power Intelligent Control) testbed developed by iTrust Labs[1]. This testbed models a real scaled-down replica of a smart-grid, capable of generating up to 72kVA [61]. This testbed is used for research and experimentation of cyber security mechanisms in the context of *Critical Infrastructures* [62]. The dataset used in this work considers the set of assets in EPIC, as well as the respective connectivities (Figure 5.1), and a business-process built based on the description of EPIC's processes (Figure 5.3).

The network architecture containing the assets of the experimental dataset, as in EPIC, is depicted in Figure 5.1. The dataset contains six types of assets — SCADA (Supervisory Control and Data Acquisition), Historian, PLCs (Programmable Logic Controllers), IEDs (Intelligent Electronic Devices), SWs (Network Switches), and APs (Access Points). Each of the assets is prefixed by a letter — C, G, M, T, S — according to the stage it belongs to — *control*, *generation*, *microgrid*, *transmission*, and *smarthome*.

Each type of asset has a different purpose, described as follows:

- The **SCADA** (Supervisory Control and Data Acquisition) Workstation, which runs the software **PCVue**[2], is responsible for monitoring and controlling the overall system operation. In particular, the SCADA is responsible for issuing commands for controlling the PLCs.
- The **Historian** stores and logs data associated with the SCADA.
- **PLCs** (Programmable Logic Controllers) are devices used to manage Industrial equipment and

[1] https://itrust.sutd.edu.sg/testbeds/electric-power-intelligent-control-epic/
[2] https://www.pcvuesolutions.com/

processes. The software that these devices run is called **CoDeSys**[3]. One particular function of these devices is relaying commands from SCADA to the IEDs, that directly control the circuit breakers of the system.

- **IEDs** (Intelligent Electronic Devices) run a software called **SIPROTEC**[4]. Their main function is directly opening and closing circuit breakers according to instructions from PLCs or if anomalies in current, voltage, or frequency are detected.
- **SWs** (Network Switches) establish connectivity between assets belonging to different subnets.
- **APs** (Access Points) function similarly to Network Switches, but they are connected to each other wirelessly.

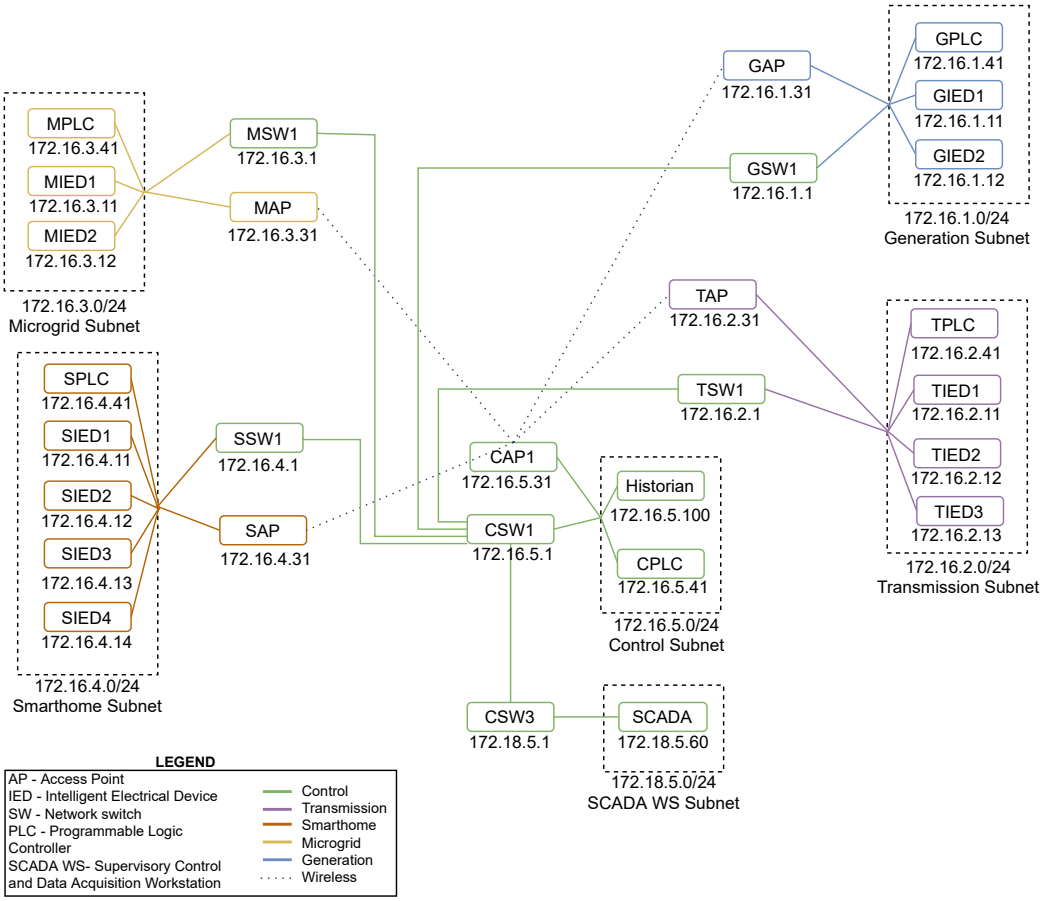


Figure 5.1: EPIC's Network Diagram.

As shown in Figure 5.1, the assets in each stage (*control*, *generation*, *microgrid*, *transmission*, and *smarthome*) are divided into separate subnets, and the *control* stage is further divided into the SCADA WS Subnet and the Control Subnet. Both wired and wireless connections are established between the subnets, the former through Network Switches and the latter through Access Points.

Besides the connectivity between the different assets, the entry-point vulnerabilities that each type of asset is susceptible to are also required, as well as their CVSS scores. Table 5.1 presents a list of

[3]https://www.codesys.com/

[4]https://new.siemens.com/global/en/products/energy/energy-automation-and-smart-grid/protection-relays-and-control/siprotec-5.html

the considered vulnerabilities for the asset types SCADA, PLC, and IED, and their CVSS v3.1[5] Scores, taken from NIST's National Vulnerability Database[6].

Table 5.1: List of vulnerabilities considered for each asset type (SCADA, PLCs and IEDs).

Type of Asset	Vulnerability ID	Vulnerability Description	STRIDE Categories	CVSS Score
SCADA WS (Win7 machine running PCVue v11)	CVE-2020-26867 (Execution of arbitrary code)	Untrusted data is deserialized without validating the result, making it possible to remotely execute arbitrary code.	Elevation of Privilege, Tampering	9.8
	CVE-2020-26868 (Denial of Service)	Unauthorized users are able to modify information used to validate messages, making it vulnerable to a denial-of-service attack.	Denial of Service	7.5
	CVE-2020-26869 (Information Disclosure)	Unauthorized users are able to access session data of legitimate users.	Information Disclosure	7.5
	CVE-2019-0752 (Remote code execution)	Memory corruption exploit in Microsoft Internet Explorer allows the attacker to remotely execute arbitrary code in the context of the current user.	Elevation of Privilege, Tampering	7.5
PLC (WAGO PFC200 running CoDeSys)	CVE-2018-5459 (Execution of unauthorized commands)	An attacker is able to execute unauthorized commands (such as manipulating the PLC application) by sending specially crafted TCP packets.	Elevation of Privilege, Tampering	9.8
IED (Siemens Relays running SIPROTEC)	CVE-2019-10938 (Execution of arbitrary code)	A vulnerability which allows the execution of arbitrary code before firmware verification.	Escalation of Privilege, Tampering	9.8
	CVE-2019-19279 (Denial of Service)	Specially crafted UDP packets can cause denial of service, recoverable only through a device reboot.	Denial of Service	7.5

In the evaluation process, it will be assumed that, by default, the attacker will choose to exploit the vulnerability CVE-2019-0752 in order to perform a *remote code execution* on the SCADA as the entry-point threat. This assumption derives from the fact that SCADA workstations are often the entry-point of cyber-attacks due to them being accessible through the Internet [63].

The diagram in Figure 5.2 presents the connections between the physical stages of the electrical grid — the *smarthome* is the load, i.e., the consumer of power. To provide power to the *smarthome*, there can be two possibilities. The first possibility is using the *generation* stage, which contains three 10kW generators and produces up to 30kW of electrical power. This power then goes through the *transmission* stage, which contains a transformer to step down the voltage to the *smarthome*. The second possibility

[5]https://www.first.org/cvss/v3.1/specification-document
[6]https://nvd.nist.gov/

is relying on the *microgrid* — the *microgrid* consists of a set of photovoltaic (PV) cells, with a maximum power of 34kW, as well as a battery bank to store excess power and an inverter to convert the direct current (DC) from the PV panels and batteries into alternating current (AC) to be used by the *smarthome*.

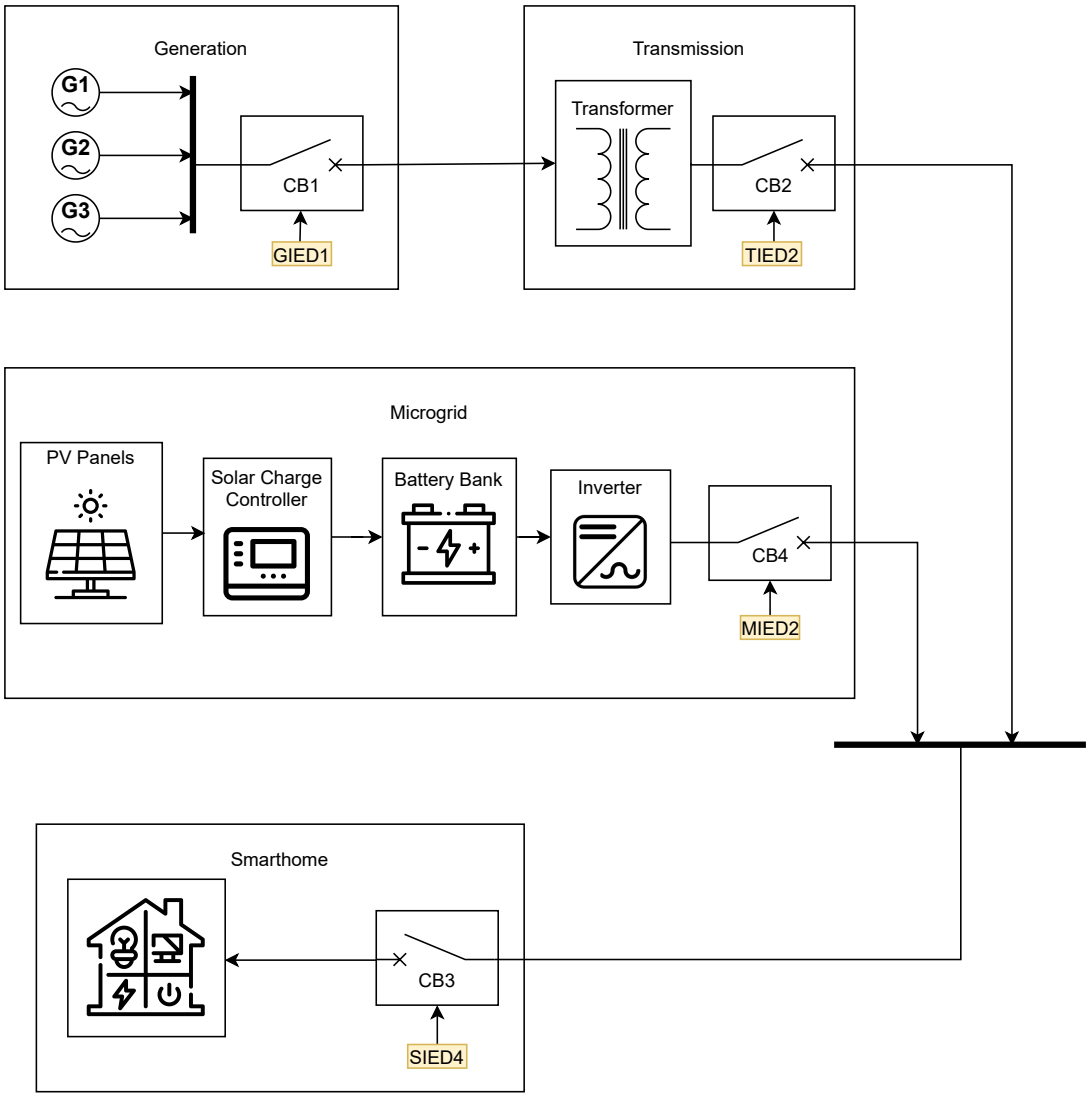


Figure 5.2: Electrical diagram of the system.

There is also the possibility of both sources (*microgrid* and *generation*) supplying power simultaneously. This is possible because the inverter used in the *microgrid* stage (Sunny Island 8.0H[7]) is able to automatically synchronize its output voltage with the grid's voltage, allowing the two sources to be connected in parallel without the risk of electrical problems occurring.

There are circuit breakers (CBs) between each of the stages, as shown in Figure 5.2, in order to control the source of power to be used by the *smarthome*. These circuit breakers are opened/closed directly by IEDs in each of the stages. However, it is from the SCADA that the commands for opening or closing the circuit breaker originates. So, in order to open or close a specific circuit breaker, the SCADA sends to the PLC responsible for that stage the open or close command and then the PLC relays this

[7] https://www.sma.de/en/products/battery-inverters/sunny-island-44m-60h-80h.html

command to the IED that controls that specific circuit breaker.

The business-process that will be considered for the evaluation will be the *power supply to the smarthome* illustrated on the diagram in Figure 5.3. Since there is a redundancy in this supply, i.e., the power can either come from the *generation* or *microgrid* stages, the business-process diagram contains an *exclusive gateway* (G1) at the beginning, whose top branch corresponds to the supply of power in grid-connected mode, i.e., from the *generation* stage, whereas the bottom branch corresponds to the supply of power from the *microgrid*. In the top branch, there is then a *parallel gateway* (G2), with each of the following branches corresponding to different circuit breakers — CB1, CB2, and CB3. This means that, in grid-connected mode, these three circuit breakers need to simultaneously be closed in order to get power from the *generation* stage to the *smarthome*. Likewise, for the power supply in *microgrid* mode (bottom branch after the *exclusive gateway* G1), both the CB4 and CB3 circuit breakers need to be closed, hence the two branches diverging from the *parallel gateway* G3.

Each of the branches corresponding to the closing of a circuit breaker contains three activities — first, the command that originates from the SCADA to the corresponding stage's PLC, then the relay of this command from the PLC to the IED, and finally the close of the circuit breaker by the IED.

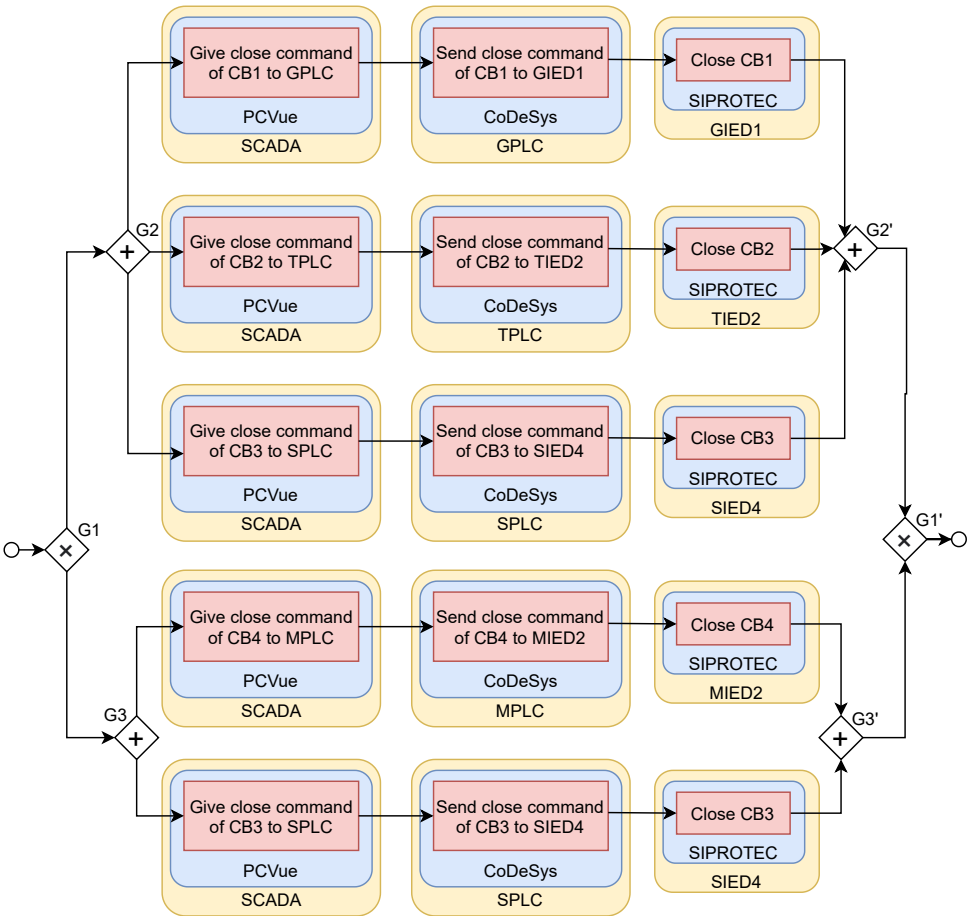


Figure 5.3: Business-process diagram of the BP *Power supply to the smarthome*.

5.2 Evaluation Process

This section has the purpose of demonstrating the experiments performed on BusICalc in order to test its capabilities and limitations. The conducted experiments were designed with the aim of answering the following questions:

1. *How does the placement of the compromised activity(ies) inside the business-process influence the impact?* (Section 5.2.1)
2. *How does the merging of trivial paths influence the impact?* (Section 5.2.2)
3. *Is BusICalc successful in identifying all the possible paths across a network?* (Section 5.2.3)
4. *How does the path taken by the attacker influence the impact?* (Section 5.2.4)
5. *How can an attacker maximize the impact delivered whilst minimizing his/her effort?* (Section 5.2.5)
6. *How does the entry-point threat influence the impact?* (Section 5.2.6)
7. *How much time does it take to compute the impact? Is the solution scalable?* (Section 5.2.7)

To answer these questions, a series of experiments were performed on the EPIC dataset, as described in the following sections.

5.2.1 Effect of impacted activities

The first experiment is aimed at studying whether BusICalc is able to produce a plausible value for the impact of a threat that is propagated through a given path, considering the significance of the affected activities to the business-process, by analysing whether the relative impact values match with what is expected. Two tests were conducted with this purpose: in the first test the path in Figure 5.4 was simulated, and in the second test the path in Figure 5.5.

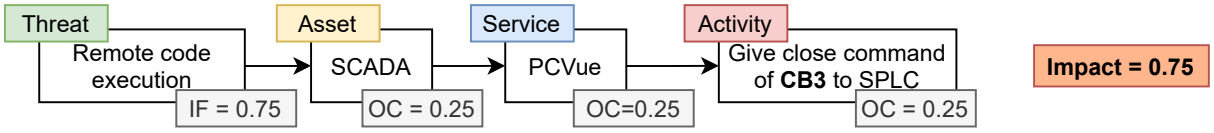


Figure 5.4: Path simulated in the first test, with *Give close command of CB3* to SPLC as the affected activity.

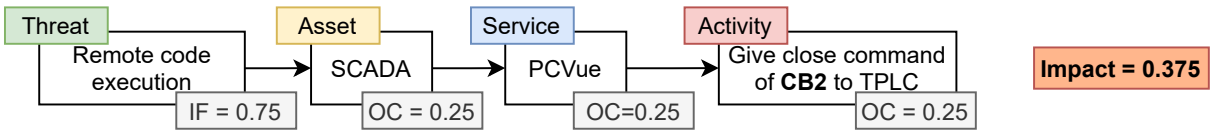


Figure 5.5: Path simulated in the second test, with *Give close command of CB2 to TPLC* as the affected activity.

In the two tests, the entry-point is a *remote code execution* threat that affects the SCADA, and subsequently the PCVue service. The difference between the paths lies in the affected activity. In the first path, the affected activity is associated with circuit breaker CB3, while in the second path, the activity is associated with CB2.

Looking at EPIC's electrical diagram (Figure 5.2), it is clear that the impact of the first path should be greater than the impact of the second path, since CB3 directly controls the supply of power to the *smarthome*, and CB2 only controls the output of power from the *transmission* stage, which means that even if CB2 becomes compromised, the supply of power to the *smarthome* is still possible through the *microgrid* stage.

This is in fact confirmed by the simulations — the impact calculated for the first path is 0.75 and for the second 0.375. These values are explained by the fact that the business-process contains two execution threads — the first thread contains all the activities above the *exclusive gateway* G1 in the business-process diagram (Figure 5.3), and the second thread contains all the activities below the *exclusive gateway* G1. The activity *Give Close Command of CB3 to SPLC* (*smarthome* PLC) belongs to both the execution threads, while the activity *Give Close Command of CB2 to TPLC* (*transmission* PLC) belongs to only one. This means that in the first path, both execution threads see their OC reduced, instead of just one in the second path. Hence, when computing the OC of the business-process (which is the average of the OCs of its execution threads (Equation 3.7)), it is natural that for the first path this value is smaller, which ultimately results in a greater value for its impact.

5.2.2 Effect of merging paths

In the second experiment, the goal is to examine the effect of the aggregation of trivial paths on the overall impact. With this purpose, a series of simulations were performed, first on a set of individual paths, and then on the aggregation of those paths. Figure 5.6 shows the trivial paths over which the simulations were performed, and Figure 5.7 shows the path that results from the merging of those trivial paths. All of the trivial paths have the same entry-point (*remote code execution*), and affect the same asset (SCADA) and service (PCVue), but each affects a different activity offered by this service. By merging the trivial paths, a path is obtained in which all the activities provided by the PCVue service are compromised.

Each of the trivial paths has an impact of either 0.375 or 0.75 (the impact of 0.75 corresponds to the path described in the previous section). The resulting impact of the merged path is 0.96. This value is greater than the impact of any of the trivial paths that comprise the merged path. In fact, this result is a property of the employed algorithm — whenever two or more paths are merged, the impact of the resulting path is greater or equal than the impact of each of the comprising paths. The proof of this property is quite straightforward: when merging two or more paths, the number of compromised activities can only either increase or remain the same. In either case, it is impossible for the OC of any activity to rise. Hence, the impact can only stay equal to the maximum of the trivial paths' impact, or rise above this value.

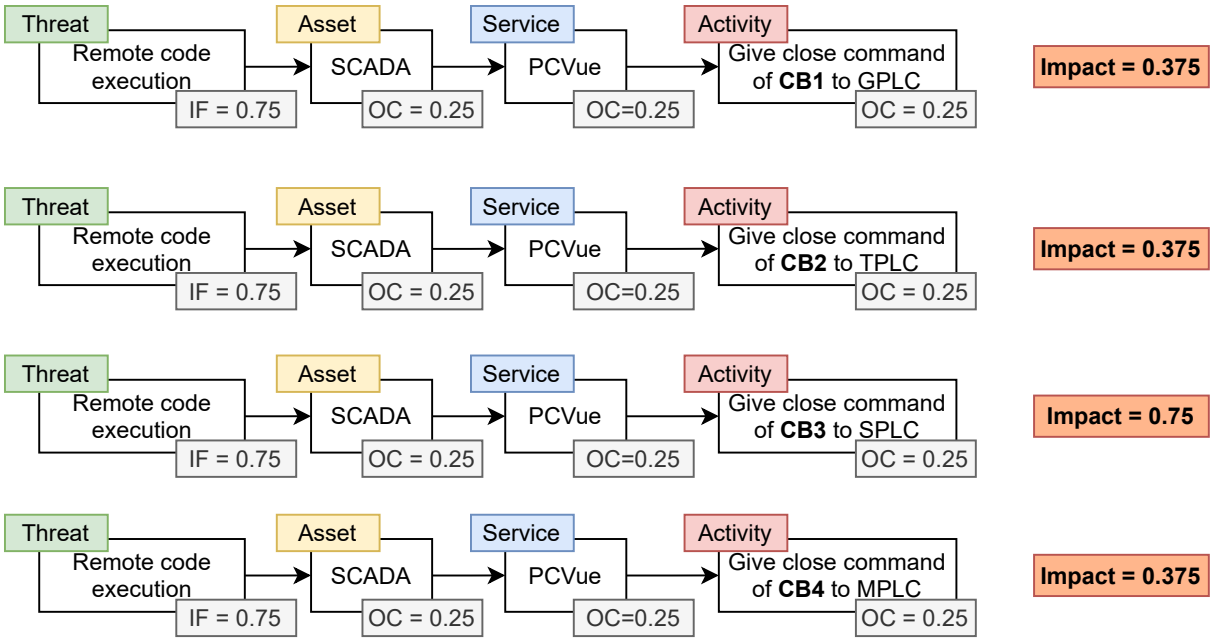


Figure 5.6: Set of four paths simulated in the first test, each affecting a different activity.

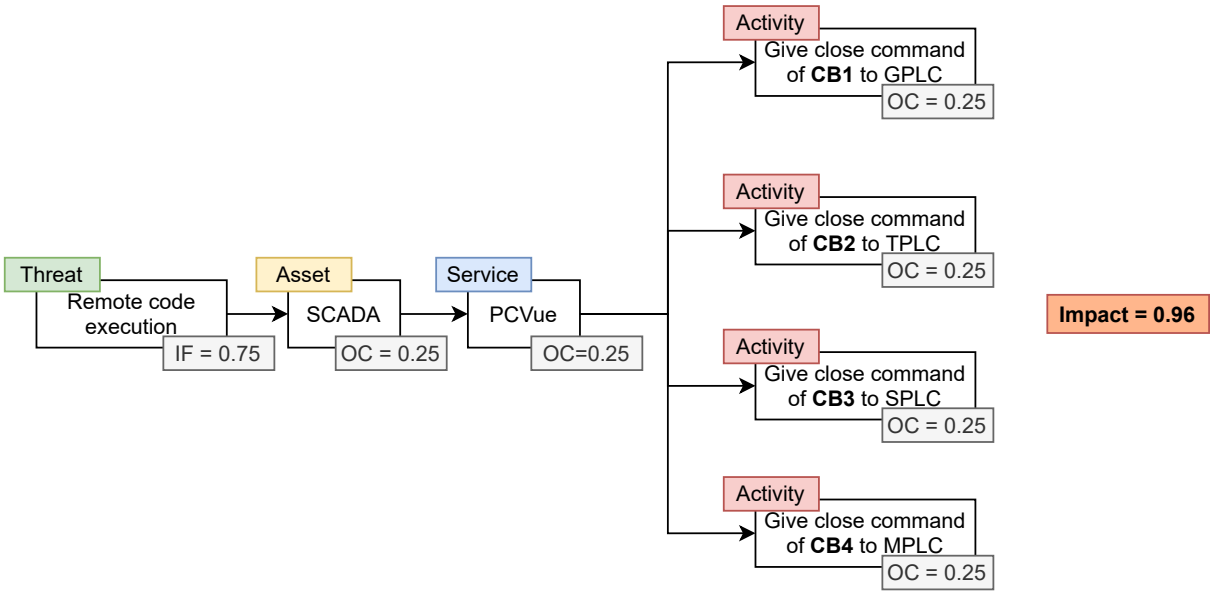


Figure 5.7: Path simulated in the second test, that results from merging the trivial paths in the previous test.

This property is verified when gradually merging the paths in Figure 5.6, until arriving at the path in Figure 5.7 — merging the first two trivial paths (both with impact of 0.375), the resulting merged path has an impact of 0.469; merging the first three trivial paths (the first two with impact of 0.375, and the third with 0.75), the resulting merged path has an impact of 0.867; and finally the merging of the four trivial paths (three of them with impact of 0.375 and one with impact of 0.75) results in the path in Figure 5.7, with impact of 0.96. This behaviour is in fact coherent with reality, since the impact of an attack that compromises several activities of an organization's business-process must take all of the affected activities into consideration, rather than, for instance, only the activity that yields the highest impact.

To further demonstrate this property, simulations were performed varying the *merge_option* parameter. The first option — *Merge Activities* — produced the paths in Figure 5.8. Since every service only provides one activity, with the exception of PCVue that provides four, all the resulting paths are trivial paths, except the first one, which has already been discussed in this section.

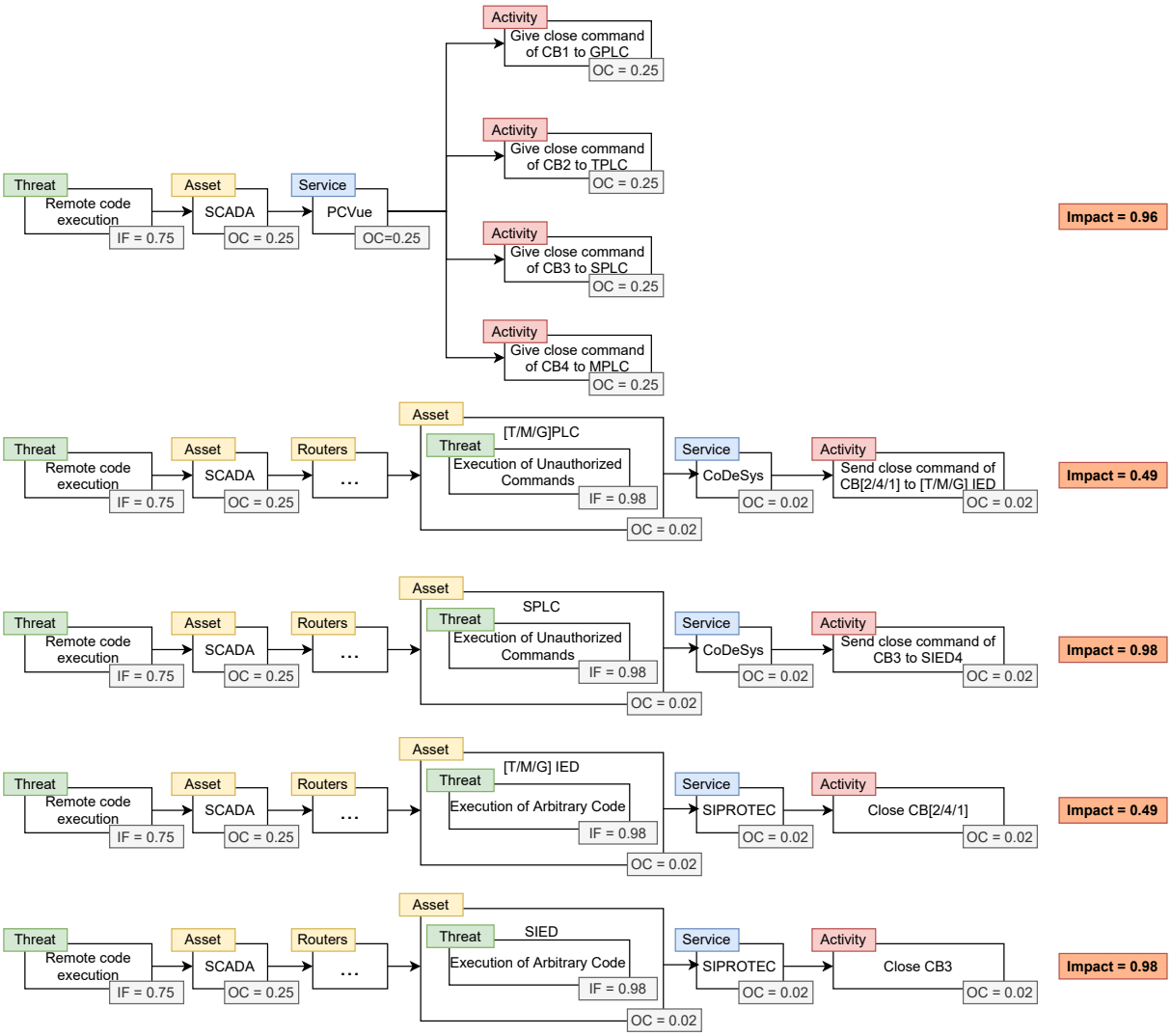


Figure 5.8: Paths that result from the merge option *Merge Activities*.

The next option — *Merge Services* — yields the exact same paths as the previous option (*Merge Activities*). The reason for this is that each asset only runs one service, so there are no further merges relative to the previous scenario.

Lastly, the option *Merge Assets* merges all the trivial paths into a single merged path (the total number of trivial paths discovered in this scenario is 55). In this example, the resulting path has an impact of approximately 1. Since this path can be seen as an aggregation of the paths that result from Figure 5.8, and the maximum impact out of all of those paths is 0.98, then the discussed property is once again verified.

5.2.3 Discovery of paths

The EPIC network used for this case study contains multiple network switches and access points, which allow for the communication between the various network assets. Subsequently, in case of an attack, these links can be leveraged by the attacker in order to amplify the inflicted impact. For this reason, it is important to be able to identify every path that can be used for that purpose. This section's aim is analysing the various paths between two assets that are identified by BusICalc.

With this goal, a simulation was made in order to determine the discovered paths between the SCADA and the GPLC. The resulting paths can be observed in Figure 5.9.

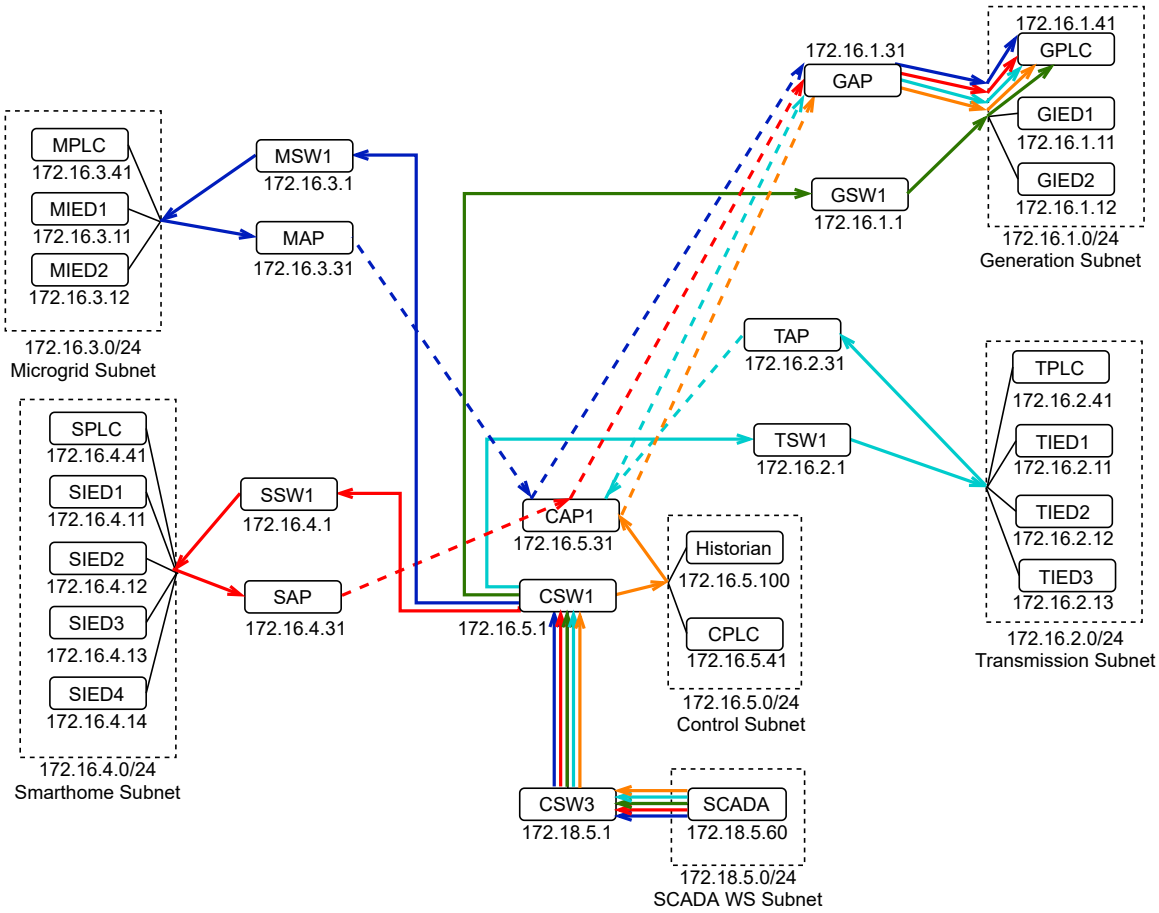


Figure 5.9: Discovered paths between the SCADA and the GPLC, each represented in a different color: dark blue, red, green, light blue, and orange.

According to this figure, a total of five paths were discovered. Among them, the green path ($SCADA \rightarrow CSW3 \rightarrow CSW1 \rightarrow GSW1 \rightarrow GPLC$) is the most direct and the only one which uses exclusively wired links. All the remaining paths use wireless links (dashed lines). More specifically, each of the remaining paths takes advantage of the fact that the access point CAP1 is a central AP that is wirelessly connected to all other APs. For example, the red path ($SCADA \rightarrow CSW3 \rightarrow CSW1 \rightarrow SSW1 \rightarrow SAP \rightarrow CAP1 \rightarrow GAP \rightarrow GPLC$) is able to reach CAP1 by entering the *smarthome* subnet (through SSW1), which contains the SAP access point that connects to CAP1. From the CAP1, the GAP can be reached, which in turn grants access to the GPLC.

Indeed this experiment shows that BusICalc is able to identify the numerous paths that exist between

two assets, which is useful to examine how those paths are distributed across the network and whether or not there are choke points between those assets (i.e., if there is any router that would cause the two assets to become disconnected if it were removed). For instance, Figure 5.9 shows that both CSW3 and CSW1 represent choke points between the SCADA and the GPLC.

5.2.4 Effect of compromised path

This section aims at analysing how the impact may vary according to the compromised path chosen.

For this experiment, two similar paths were simulated — path in Figure 5.10 and path in Figure 5.11. These two paths may not seem very similar at first look, but the activities compromised by each one are in fact equivalent. This means that if the activity *Give Close Command of CB1 to GPLC* on the first path has an OC of x, and the activity *Send Close Command of CB1 to GIED1* also has the same OC of x, then the two paths will have the same impact.

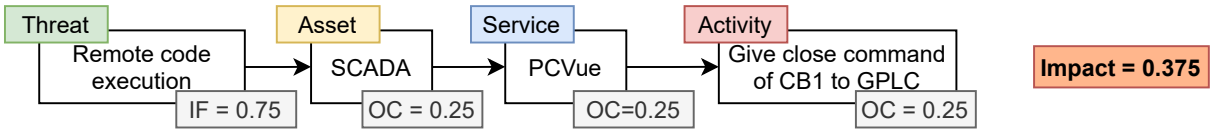


Figure 5.10: Path simulated in the first test, only affecting the Asset SCADA.

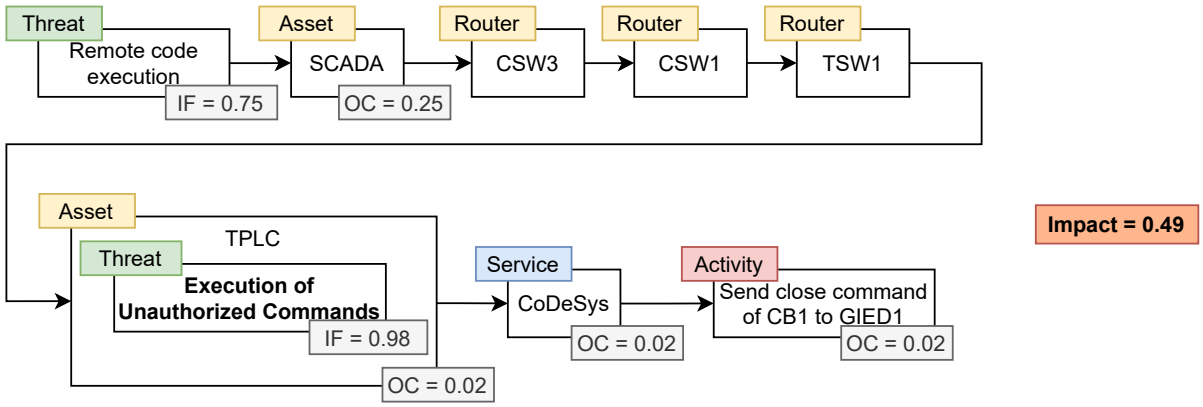


Figure 5.11: Path simulated in the second test, affecting the Asset TPLC besides the SCADA.

The actual difference between the two paths that results in them having different impacts lies in the threat exploited in the second path. While the first path is the straightforward path already discussed in previous sections, in the second path the attacker leverages the connectivity of the SCADA in order to move laterally across the network (through routers CSW3, CSW1 and TSW1) until the asset TPLC is reached. In the TPLC, the attacker finds that there is a new threat — *execution of unauthorized commands*, with an Impact Factor of 0.98. Since this new threat has an Impact Factor greater than the original entry-point threat (which has an Impact Factor of 0.75), it is assumed that the attacker will choose to exploit this new threat in order to increase the yielded impact on the organization (this behaviour is modelled by Equation 3.3 in the algorithm).

This exploitation, in turn, causes the OC of the TPLC to decrease below the OC of the SCADA. This ultimately results in the OCs of the service and activity exploited in the second path to be lower than the

OCs of the service and activity of the first path, which means that the second path will present a greater value for the impact (0.49) when compared to the impact of the first path (0.375).

5.2.5 Impact/attacker effort balance

This section has the goal of demonstrating how the behavior of an attacker can be predicted by leveraging BusICalc.

It is a fair assumption that an attacker will try to maximize the impact inflicted on the business organization while at the same time minimizing his/her effort. The effort of an attacker can be modeled by the number of trivial paths exploited.

One way to determine which path the attacker will take is limiting the number of trivial paths to a reasonable amount and then determining the path with the highest impact.

For example, suppose the attacker is only willing to exploit up to three trivial paths. In this case, the most likely path taken by the attacker would be the path composed of three trivial paths that yield the highest impact. In order to determine this path, one option is building an algorithm to simulate all paths in these circumstances. However, in this section, this path will be determined through a logical investigation.

As argued in Section 5.2.1, the activities whose exploitation has a greater impact are the activities associated with circuit breaker CB3. Hence, the trivial paths that maximize the impact will each contain one of those activities — *Give close command of CB3 to SPLC*, *Send close command of CB3 to SIED4* and *Close CB3*. This means, necessarily, that the first trivial path compromises the SCADA and PCVue; the second exploits the SPLC and CoDeSys; and the third compromises the SIED4 and SIPROTEC.

The only missing link is now the entry-point. In Section 5.2.4, it was observed that the impact of a path is greater when the Impact Factor of the exploited threat increases. In this case, since the most critical threat of each asset has the same Impact Factor (0.98), then the entry-point asset can be any of the three. If, on the other hand, one of the assets had a threat whose Impact Factor was greater than any of the other assets' threats, then that pair (asset, threat) would have to be chosen as the entry-point.

In this case, suppose the SCADA is chosen as the entry-point asset. In order to maximize the impact, the chosen entry-point threat will be *execution of arbitrary code*, since it has the highest Impact Factor (0.98). Since it is possible to reach the two other mentioned assets from the SCADA, the path is now complete. Figure 5.12 presents the final merged path composed by the three trivial paths.

This section has shown that an attack path can be carefully crafted in order to obtain an impact very close to the maximum possible value ($0.999992 \approx 1$), by compromising only a limited number of assets/services/activities. For comparison, the merge option *Merge Assets* discussed in Section 5.2.2 managed to obtain an impact value only slightly higher (approximately 1), by merging the total of 55 trivial paths discovered.

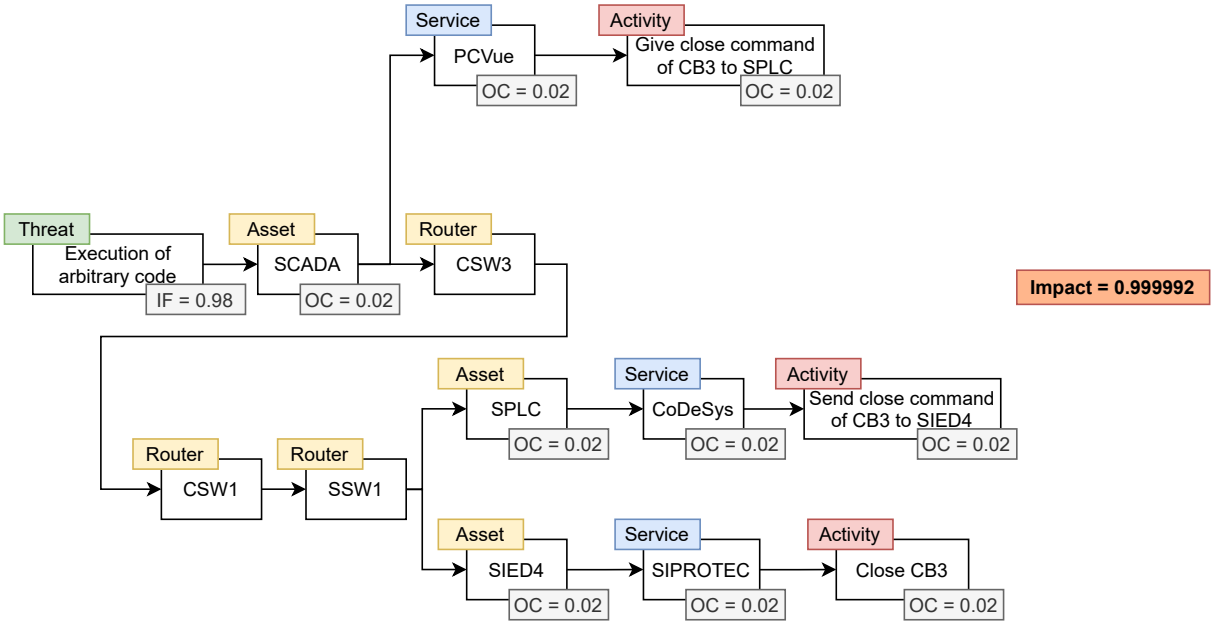


Figure 5.12: Merged path that maximizes impact, composed of three trivial paths.

5.2.6 Effect of Entry-point Threat

In this section, the goal is to study the effect that the chosen entry-point threat might have on the impact. Thus, a comparison was made between two similar paths that differ only on the user-chosen entry-point threat.

The entry-point threat of the path in Figure 5.13 is a *remote code execution*, which has an Impact Factor of 0.75, while the entry-point threat of the path in Figure 5.14 is an *execution of arbitrary code*, which has a higher Impact Factor of 0.98. Both paths affect the same asset, service, and activity.

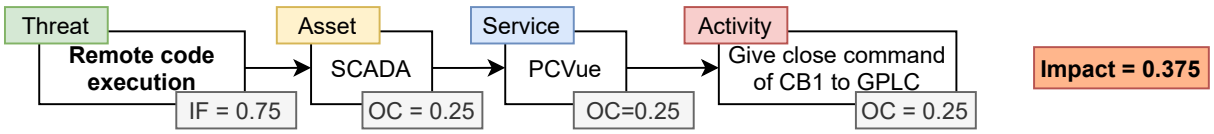


Figure 5.13: Path simulated in the first test, with the entry-point threat *remote code execution*.

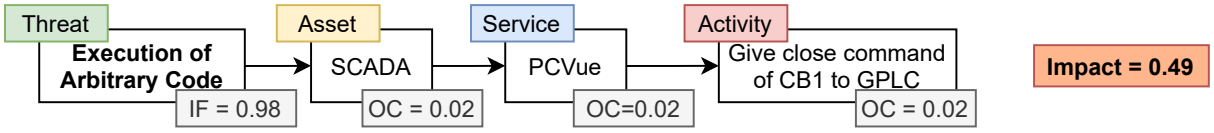


Figure 5.14: Path simulated in the second test, with the entry-point threat *execution of arbitrary code*.

It is to be expected that the path whose threat has a higher Impact Factor should have a higher impact, since that Impact Factor is used to determine the propagation of operationality loss through the affected assets/services/activities.

In fact, this is exactly what is observed in this experiment: the first path, which has a lower threat Impact Factor, also has a lower impact (0.375) compared to the second path, which has a greater threat Impact Factor and a resulting impact of 0.49.

The effect of the entry-point threat can be studied even more thoroughly by performing a set of simulations with varying Impact Factor for the entry-point threat. Thus, for some of the paths shown previously, a set of simulations was performed with varying Impact Factor for the entry-point threat, from 0 to 1 with a step of 0.01 (Figures 5.15, 5.16, 5.17, 5.18 (a), and 5.19 (a)), and furthermore from 0.97 to 1 with a step of 0.001 in the cases where the first set of experiments was not conclusive (Figures 5.18 (b), and 5.19 (b)).

The first path in which the variation of IF was studied is presented in this section, in Figure 5.13. Figure 5.15 shows the variation of the impact of this path with the Impact Factor of the entry-point threat. Confirming what was previously observed, this figure shows that the impact of the path increases as the Impact Factor increases. Moreover, it also shows that this relationship is linear. The reason for this is that this is a trivial path that affects only one asset, which means that the OCs of the asset, service and activity are directly derived from the Impact Factor of the entry-point threat. As a result, the impact of the path is proportional to the Impact Factor.

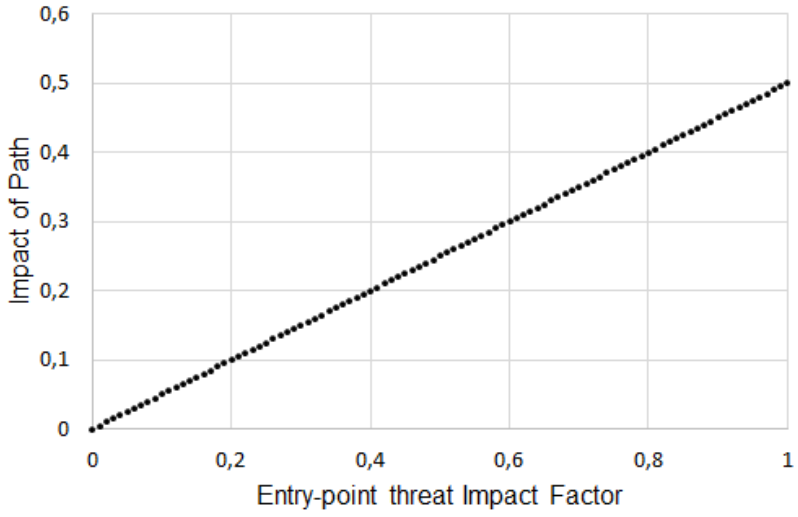


Figure 5.15: Relationship between Impact Factor of the entry-point threat and impact of the path in Figure 5.13.

Likewise, for the path in Figure 5.4, the relationship between impact and Impact Factor is also linear (as shown in Figure 5.16). However, as this path affects the activity *Give close command of CB3 to SPLC* instead of *Give close command of CB1 to GPLC*, instead of ranging between 0 and 0.5, it ranges between 0 and 1. This is due to the fact that the activities involving CB3 are more critical than the activities that involve other Circuit Breakers, as explained in Section 5.2.1.

The next path to be evaluated is the path in Figure 5.7. This path differs from the previous two because it no longer compromises a single activity. Instead, it compromises a total of four activities, all provided by the same service and asset. As a result, Figure 5.17 shows that the relationship between the Impact Factor of the entry-point threat and the impact of the path is no longer linear, but instead polynomial. This happens because the OC of each affected activity is derived from the IF of the entry-point threat, as in the previous paths, but in order to compute the impact, these activities' OCs are multiplied by each other according to Equation 3.6, which results in a polynomial relationship between

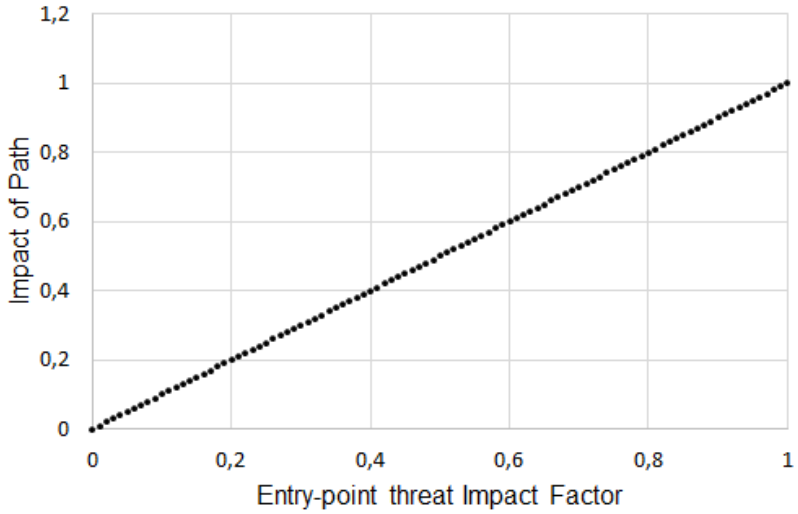


Figure 5.16: Relationship between Impact Factor of the entry-point threat and impact of the path in Figure 5.4.

IF and impact.

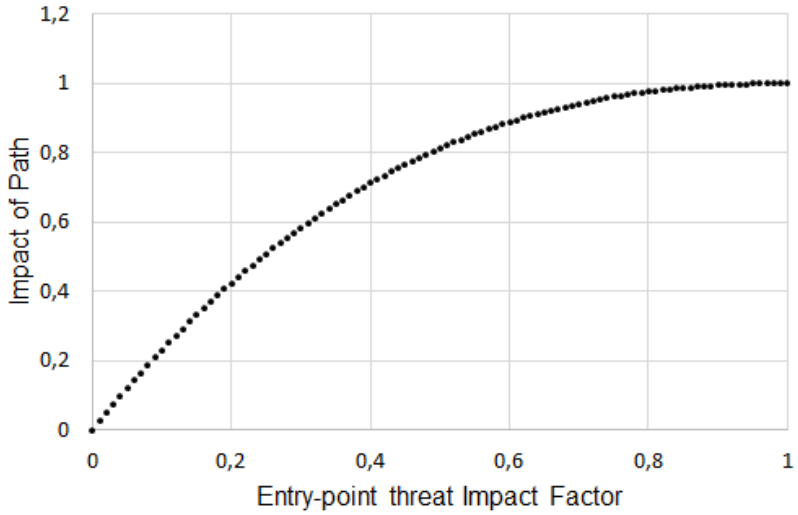
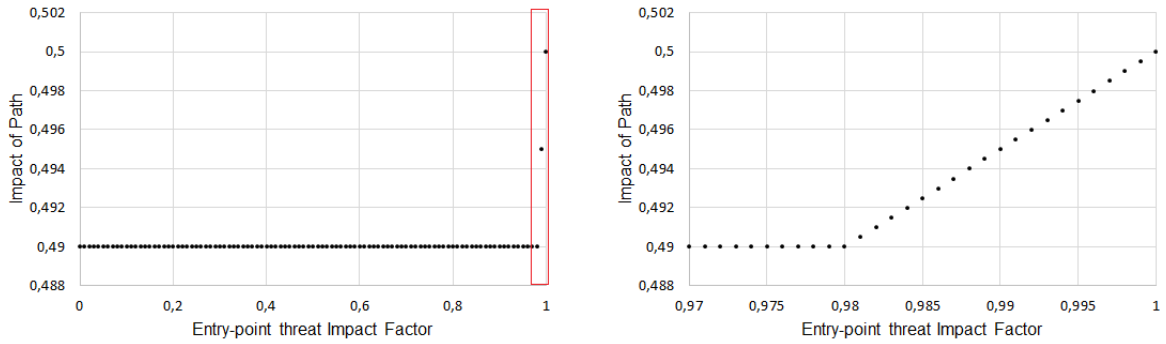


Figure 5.17: Relationship between Impact Factor of the entry-point threat and impact of the path in Figure 5.7.

Next, the path in Figure 5.11 was evaluated using this method. The resulting relationship between the Impact Factor of the entry-point threat and the impact of the path is presented in Figure 5.18. According to this figure, the impact of the path is constant and equal to 0.49 for IF lower or equal than 0.98, and then increases linearly with IF, from 0.49 to 0.5, when the IF changes from 0.98 to 1. To understand this behaviour, it is important to comprehend this path. This is a trivial path in which the attacker moves from the entry-point asset (SCADA) to another asset (TPLC). The TPLC contains a threat — *execution of unauthorized commands* — with Impact Factor of 0.98. This means that, if the entry-point threat has a lower Impact Factor than 0.98, the attacker will choose to exploit the threat in the TPLC, which yields a higher impact (as explained in Section 5.2.4), and hence the IF of the entry-point threat will not affect the final impact of the path, which is why Figure 5.18 shows a constant impact of 0.49 for an IF of the entry-

point threat lower than 0.98. On the other hand, if the entry-point threat has a higher IF than the threat in the TPLC, then the attacker will not exploit the threat in the TPLC, and the OCs of the assets, service and activity will be directly derived from the IF of the entry-point threat, resulting in a linear relationship between the impact of the path and the entry-point IF.



(a) Impact Factor between 0 and 1

(b) Impact Factor between 0.97 and 1

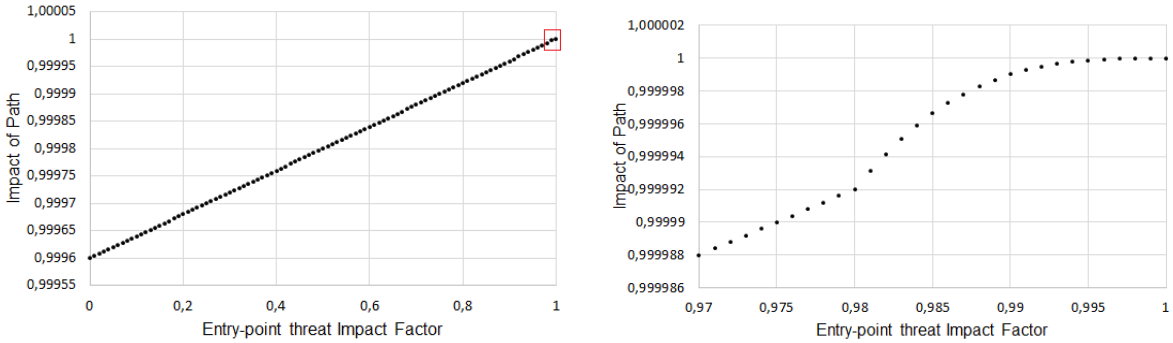
Figure 5.18: Relationship between Impact Factor of the entry-point threat and impact of the path in Figure 5.11, where Figure (b) presents in detail the area marked in red in Figure (a).

The last path to be evaluated using this method is the path in Figure 5.12. This path compromises three assets — SCADA, SPLC and SIED4 — where the SCADA is the entry-point asset, and through each of the assets one service and one activity are affected. Figure 5.19 presents the variation of the impact of this path with the Impact Factor of the entry-point threat. The first observation from this figure is that the impact of the path is not zero when the IF of the entry-point threat is zero. This happens because both the SPLC and the SIED4 are affected by threats different than the entry-point threat (with Impact Factors equal to 0.98 each), which means that even with the IF of the entry-point threat equal to zero, the two activities provided by the SPLC and SIED4 will not be fully operational (which results in an impact of 0.9996 for the path). As the IF of the entry-point threat increases from 0 to 0.98, the OCs of the two activities provided by the SPLC and the SIED4 remain constant, since this IF is lower than the IFs of the threats that directly affect these assets. Hence, only the activity provided by the SCADA will depend on the IF of the entry-point threat, making the impact of the path linearly dependent on the IF of the entry-point threat. When the Impact Factor of the entry-point threat is greater than 0.98, the OCs of the three activities start depending on this IF, which results in a polynomial relationship between the Impact Factor and the impact of the path.

5.2.7 Performance Evaluation

This section is aimed at evaluating the performance of BusICalc, by measuring and analysing its execution time in different situations. For that reason, several simulations were performed, with a network of varying size.

The model network used for these simulations is presented in Figure 5.20. According to this model, there are N subnets, each with K assets. All subnets are connected to the same router, which implies that every asset is able to reach every other asset in the network. There are also K services. Each of



(a) Impact Factor between 0 and 1

(b) Impact Factor between 0.97 and 1

Figure 5.19: Relationship between Impact Factor of the entry-point threat and impact of the path in Figure 5.12, where Figure (b) presents in detail the area marked in red in Figure (a).

these services is run by a total of N assets, each belonging to a different subnet. Finally, there is a single business-process with a total of K activities. These activities are executed sequentially in the business-process. Each of the activities is provided by a different service, and each service only provides one activity.

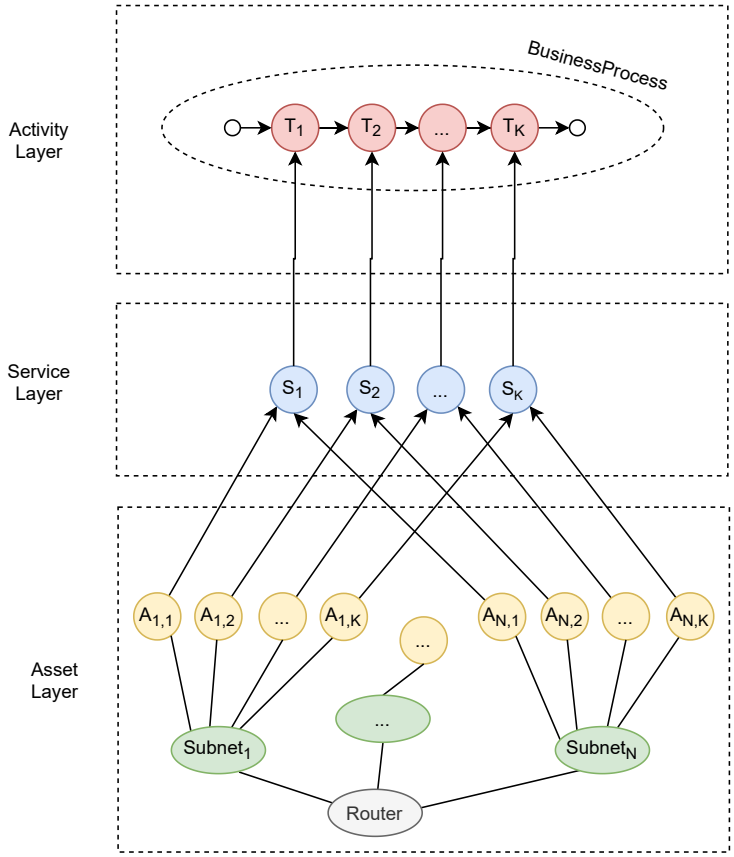


Figure 5.20: Network model used to test performance.

By analysing this network, it is possible to observe that there is a total of $N \times K$ assets (N subnets with K assets each). It is also possible to infer that the total amount of trivial paths is $N \times K$ as well — by choosing, for instance, $A_{1,1}$ as the entry-point asset, there is a trivial path $A_{1,1} \rightarrow S_1 \rightarrow T_1$, and there are an additional $N \times K - 1$ trivial paths, each affecting a different asset (e.g., $A_{1,1} \rightarrow Subnet_1 \rightarrow$

$A_{1,2} \rightarrow S_2 \rightarrow T_2$).

BusICalc was simulated with this network using different values for K and N (1, 2, 5, 7 and 10). The selected entry-point was asset $A_{1,1}$, and the merge option selected was *Merge Assets*, which aggregates all the trivial paths into a single merged path. This option constitutes the worst-case scenario in terms of computational cost. For each of these simulations, the execution time for each of the modules that comprise BusICalc was measured.

From these simulations, it was concluded that the majority of the total computational time is spent waiting for the processing of BIA — on average, the Impact Calculation Module only took around 0.027% of the total time of a simulation, while the remaining 99.973% was spent on the Setup Module, of which 81.5% of the time, on average, was spent in the processing of BIA. This considerable difference between the execution times of the different modules can be partially explained by the fact that the two sub-modules *setup.py* and *populate_network.py*, inside the Setup Module, make remote calls to BIA and to BIA's database, which are subject to significant computational overhead.

In order to evaluate whether the solution for impact calculation is scalable, it is necessary to analyse the relationship between the execution time of the Impact Calculation Module, and the number of assets in the network. Figure 5.21 presents the results obtained from the simulations, in which the vertical axis is the execution time of the Impact Calculation Module in each simulation (in milliseconds), and the horizontal axis is the number of assets of the network in each simulation ($K \times N$), which in this case is equal to the number of trivial paths discovered.

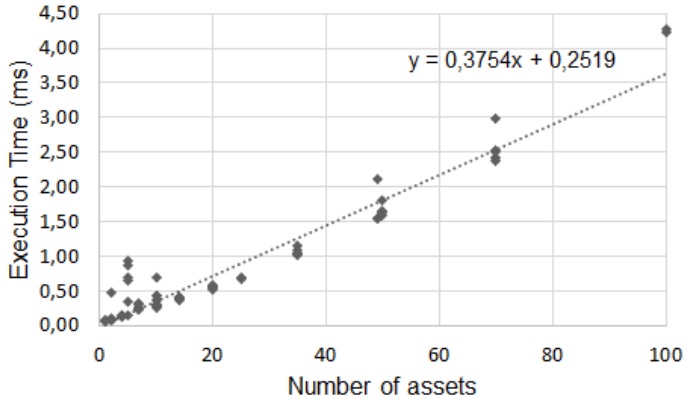


Figure 5.21: Relationship between the number of assets of the network and the execution time of the Impact Calculation Module.

This figure shows that there is a linear relationship between the number of assets and the execution time of the Impact Calculation Module. This means that the solution is in fact scalable, since the required time to run the impact calculation algorithm only increases by a constant amount for each asset that is added to the network, which means that the algorithm has a complexity of $O(n)$, where n is the number of assets in the network.

5.3 Summary

This chapter has presented the evaluation process designed to test BusICalc.

The first section (Section 5.1) describes the EPIC dataset used for the evaluation, namely the network's topology, the physical setup of the system, and the created business-process.

The second section (Section 5.2) presents the actual simulations performed and the drawn conclusions. The most relevant results of this process can be summarized as follows:

- BusICalc is able to take into consideration the physical characteristics of the system in order to yield a sensible value for the impact (Section 5.2.1). This is possible because these characteristics are modeled into the business-process diagram, which in turn is used by the algorithm that computes the impact.
- BusICalc is also successful in modeling the impact in accordance with the relative severity of the entry-point threat exploited (Section 5.2.6), as well as with the exploited threats along a given path (Section 5.2.4).
- It was also observed that, in the cases where there are multiple paths between two assets, BusICalc was effective in discovering them (Section 5.2.3). This feature is especially useful in determining whether the connection between the two assets can be cut off by simply removing one access point/network switch.
- The option of merging attack paths becomes useful to simulate custom paths that, in the opinion of the user, might be more likely to be exploited by a real-world attacker. This allows the user to, among other things, obtain an estimate for the impact of an attack on the aggregate path and compare it with the impact of each individual path that comprises the merged path (Section 5.2.2); and check the effort that an attacker needs in order to inflict such an impact on the organization (Section 5.2.5).
- The algorithm developed for impact calculation has been shown to have a computational overhead linearly dependent on the number of assets of the network (Section 5.2.7), which means that BusICalc is able to maintain a reasonable efficiency as the size of the network increases.

Chapter 6

Conclusions

With the goal of providing a methodology capable of quantifying the impact caused by a given cyber-threat in an organization, this dissertation has studied methods that tackle impact propagation and assessment of cyber-threats, from which two methods have proven especially useful (BIA [11, 12] and Jakobson [42]). Also, the study of methods regarding *Cascading Effects* has emphasized the advantage that this methodology can have in supplying impact metrics to be used in the simulations of *Cascading Failures*.

The BIA (*Business Impact Assessment*) methodology was used as a starting point to achieve the desired functionalities. BIA is a methodology for impact assessment that is able to (1) profile an organization using a four-layer model, which includes cyber-threats, assets, services, and business-process activities; and (2) perform simulations of threat propagation paths across the modelled organization. The main feature missing from BIA is the capability of quantifying the impact of a cyber-threat propagation. The developed approach — BusICalc (*Business Impact Calculator*) — solves this issue by implementing an impact quantification algorithm, based on the work proposed by Jakobson [42]. This algorithm uses a similar four-layer model as BIA, which makes it appropriate to use in this context. As a result, BusICalc is able to simulate the propagation of a user-selected cyber-threat across an organization's network to determine which business-process activities have been affected, and subsequently assign an impact value to that scenario.

In order to test BusICalc's efficacy, a set of experiments were conducted, in which the considered testbed was modelled after a smart-grid. These experiments have shown that BusICalc is capable of producing coherent impact metrics for distinct situations that consider different sets of attack paths and exploited threats, ultimately proving it successful in delivering its primary objective. The experiments have also shown that the developed proof-of-concept is scalable, since it has a complexity of $O(n)$, where n represents the size of the dataset.

6.1 Achievements

With the development of BusICalc, this work has accomplished the following achievements:

- Creation of a scalable tool, capable of quantifying the impact delivered by a simulated attack on the critical business-processes of an organization, with the option of configuring the simulation to better replicate the attacker's behaviour. Additionally, the tool can help in identifying the most impactful threats, that should be considered in the organization's risk management procedure.
- Demonstration that BusICalc is successful in calculating the impact of cyber-threats on physical processes (in this case, delivery of power to a specific section of a smart-grid).
- Although having been evaluated in the context of a smart-grid, BusICalc can be applied to other organization domains (e.g., business corporations, other *Critical Infrastructures*, armed forces), since it was designed independently from this constraint.

6.2 Future Work

The accomplishments that resulted from the development of BusICalc would further benefit from the exploration of the following points:

- The study of *Cascading Failures* between different organizations/infrastructures could be improved by modelling the physical components of the organizations (e.g., Circuit Breakers, Power Lines, Generators, Pumps, Valves, Motors, Sensors), and their respective interdependencies, which would allow the simulation of the propagation of failures among the interconnected organizations. For instance, the original EPIC testbed supplies power to a water treatment plant — Secure Water Treatment (SWaT) — and to a water distribution system — Water Distribution (WADI). By modelling the dependency of the physical components of these systems (e.g., pumps, valves) on the energy supplied by EPIC, it would be possible to simulate the *Cascading Effects* that result from a failure in the supply of power.
- The algorithm could evolve in order to programmatically determine the attack path with the maximum possible inflicted impact, considering the attacker has limited resources (e.g., he/she is only willing to exploit up to x assets), as was described in Section 5.2.5.
- Refining the rules that govern the threat propagation simulations. For instance, currently, the rules assume that an attacker with access to a compromised asset is able to compromise the remaining assets connected to it. Realistically, not all types of threats would grant the attacker access to the neighbouring assets.

Bibliography

[1] W. Hurst, N. Shone, and Q. Monnet. Predicting the effects of DDoS attacks on a network of critical infrastructures. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1697–1702. IEEE, 2015.

[2] S. M. Rinaldi. Modeling and simulating critical infrastructures and their interdependencies. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 8–pp. IEEE, 2004.

[3] T. Grafenauer, S. König, S. Rass, and S. Schauer. A simulation tool for cascading effects in interdependent critical infrastructures. In *Proceedings of the 13th International Conference on availability, reliability and security*, pages 1–8, 2018.

[4] A. Cherepanov and R. Lipovsky. Industroyer: Biggest threat to industrial control systems since stuxnet. *WeLiveSecurity, ESET*, 12, 2017.

[5] 5 Million Americans Have Lost Power From Texas to North Dakota After Devastating Winter Storm. `https://time.com/5939633/texas-power-outage-blackouts/`, . Accessed: 24-03-2021.

[6] Texas governor says he is responsible for the status of ERCOT, vows reforms. `https://edition.cnn.com/2021/02/18/weather/texas-winter-storm-thursday`, . Accessed: 24-03-2021.

[7] Number of Texas deaths linked to winter storm grows to 151, including 23 in Dallas-Fort Worth area. `https://www.dallasnews.com/news/weather/2021/04/30/number-of-texas-deaths-linked-to-winter-storm-grows-to-151-including-23-in-dallas-fort-worth-area/`, . Accessed: 27-05-2021.

[8] Texas winter storm costs could top $200 billion — more than hurricanes Harvey and Ike. `https://www.cbsnews.com/news/texas-winter-storm-uri-costs/`, . Accessed: 27-05-2021.

[9] T. Maynard and N. Beecroft. Business Blackout: The Insurance Implications of a Cyber-Attack on the US Power Grid. *Lloyd's of London. Accessed March*, 15:2019, 2015.

[10] Y. Y. Haimes and P. Jiang. Leontief-based model of risk in complex interconnected infrastructures. *Journal of Infrastructure systems*, 7(1):1–12, 2001.

[11] O. S. B. de Carvalho. BP-IDS - Attack Impact Assessment. Master's thesis, Instituto Superior Técnico, 2020.

[12] C. Köpke, K. Srivastava, L. König, N. Miller, M. Fehling-Kaschek, K. Burke, M. Mangini, I. Praça, A. Canito, O. Carvalho, et al. Impact Propagation in Airport Systems. *Cyber-Physical Security for Critical Infrastructures Protection*, 12618:191, 2020.

[13] Z. Huang, C. Wang, A. Nayak, and I. Stojmenovic. Small cluster in cyber physical systems: Network topology, interdependence and cascading failures. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2340–2351, 2014.

[14] L. Che, X. Liu, T. Ding, and Z. Li. Revealing impacts of cyber attacks on power grids vulnerability to cascading failures. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(6):1058–1062, 2018.

[15] R. Kinney, P. Crucitti, R. Albert, and V. Latora. Modeling cascading failures in the North American power grid. *The European Physical Journal B-Condensed Matter and Complex Systems*, 46(1): 101–107, 2005.

[16] W. Wang, S. Yang, F. Hu, H. E. Stanley, S. He, and M. Shi. An approach for cascading effects within critical infrastructure systems. *Physica A: Statistical Mechanics and its Applications*, 510:164–177, 2018.

[17] Y. Wang and F. Zhang. Modeling and analysis of under-load-based cascading failures in supply chain networks. *Nonlinear Dynamics*, 92(3):1403–1417, 2018.

[18] Q. Yang, C. M. Scoglio, and D. M. Gruenbacher. Robustness of supply chain networks against underload cascading failures. *Physica A: Statistical Mechanics and its Applications*, 563:125466, 2021.

[19] Y. Zeng and R. Xiao. Modelling of cluster supply network with cascading failure spread and its vulnerability analysis. *International Journal of Production Research*, 52(23):6938–6953, 2014.

[20] D. M. Lambert and M. C. Cooper. Issues in supply chain management. *Industrial marketing management*, 29(1):65–83, 2000.

[21] Cybersecurity and Infrastructure Security Agency. Critical Infrastructure Sectors. `https://www.cisa.gov/critical-infrastructure-sectors`, 2020. Accessed: 14-04-2021.

[22] G. H. Baker and S. Volandt. Cascading consequences: Electrical grid critical infrastructure vulnerability. *DomPrep Journal*, 14(5):10–18, 2018.

[23] D. Rehak, P. Senovsky, M. Hromada, T. Lovecek, and P. Novotny. Cascading impact assessment in a critical infrastructure system. *International journal of critical infrastructure protection*, 22:125–138, 2018.

[24] I. B. Utne, P. Hokstad, and J. Vatn. A method for risk modeling of interdependencies in critical infrastructures. *Reliability Engineering & System Safety*, 96(6):671–678, 2011.

[25] A. E. Motter and Y.-C. Lai. Cascade-based attacks on complex networks. *Physical Review E*, 66 (6):065102, 2002.

[26] L. Duenas-Osorio and S. M. Vemuru. Cascading failures in complex infrastructure systems. *Structural safety*, 31(2):157–167, 2009.

[27] C. Köpke, J. Schäfer-Frey, E. Engler, C. P. Wrede, and J. Mielniczek. A joint approach to safety, security and resilience using the functional resonance analysis method. In *8th REA Symposium on Resilience Engineering: Scaling up and Speeding up*, 2020.

[28] O. H. Ramirez Agudelo, C. Köpke, and F. Sill Torres. Bayesian Network Model for Accessing Safety and Security of Offshore Wind Farms. 2020.

[29] V. R. Palleti, S. Adepu, V. K. Mishra, and A. Mathur. Cascading effects of cyber-attacks on interconnected critical infrastructure. *Cybersecurity*, 4(1):1–19, 2021.

[30] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE control systems magazine*, 21(6):11–25, 2001.

[31] D. Firesmith. System Resilience: What Exactly is it? `https://insights.sei.cmu.edu/sei_blog/2019/11/system-resilience-what-exactly-is-it.html`. Accessed: 24-03-2021.

[32] W. Leontief. *Input-output economics*. Oxford University Press, 1986.

[33] C. Cao, L.-P. Yuan, A. Singhal, P. Liu, X. Sun, and S. Zhu. Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 330–348. Springer, 2018.

[34] E. M. Hutchins, M. J. Cloppert, R. M. Amin, et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[35] MITRE ATT&CK. `https://attack.mitre.org/`, . Accessed: 16-04-2021.

[36] B. E. Strom, J. A. Battaglia, M. S. Kemmerer, W. Kupersanin, D. P. Miller, C. Wampler, S. M. Whitley, and R. D. Wolf. Finding cyber threats with ATT&CK-based analytics. *The MITRE Corporation, Bedford, MA, Technical Report No. MTR170202*, 2017.

[37] T. Smith. MITRE ATT&CK Navigator Categorization. `https://github.com/TravisFSmith/mitre_attack`. Accessed: 16-04-2021.

[38] MITRE ATT&CK Navigator. `https://mitre-attack.github.io/attack-navigator/`, . Accessed: 16-04-2021.

[39] MITRE CALDERA. `https://github.com/mitre/caldera`, . Accessed: 16-04-2021.

[40] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6):85–89, 2006.

[41] R. E. Sawilla and X. Ou. Identifying critical attack assets in dependency attack graphs. In *European Symposium on Research in Computer Security*, pages 18–34. Springer, 2008.

[42] G. Jakobson. Mission cyber security situation assessment using impact dependency graphs. In *14th international conference on information fusion*, pages 1–8. IEEE, 2011.

[43] I. V. Kotenko and E. Doynikova. Evaluation of Computer Network Security based on Attack Graphs and Security Event Processing. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, 5 (3):14–29, 2014.

[44] C. Liu, A. Singhal, and D. Wijesekera. A layered graphical model for mission attack impact analysis. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 602–609. IEEE, 2017.

[45] N. Kheir, A. R. Mahjoub, M. Y. Naghmouchi, N. Perrot, and J.-P. Wary. Assessing the risk of complex ICT systems. *Annals of Telecommunications*, 73(1):95–109, 2018.

[46] Y. Sun, T.-Y. Wu, X. Liu, and M. S. Obaidat. Multilayered impact evaluation model for attacking missions. *IEEE Systems Journal*, 10(4):1304–1315, 2014.

[47] S. Noel, S. Jajodia, L. Wang, and A. Singhal. Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 1(1):135–147, 2010.

[48] N. Poolsappasit, R. Dewri, and I. Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2011.

[49] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graphs. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 698–703. IEEE, 2008.

[50] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM workshop on Quality of protection*, pages 23–30, 2008.

[51] NVD CVSS. `https://nvd.nist.gov/vuln-metrics/cvss`. Accessed: 25-03-2021.

[52] Common Vulnerability Scoring System v3.1: Specification Document. `https://www.first.org/cvss/v3.1/specification-document`. Accessed: 25-03-2021.

[53] Vulnerability Severity Using CVSS. `https://insights.sei.cmu.edu/blog/vulnerability-severity-using-cvss/`. Accessed: 25-03-2021.

[54] M. R. Fuentes and N. Huq. Securing connected hospitals, 2018.

[55] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-based Network Security Analyzer. In *USENIX security symposium*, volume 8, pages 113–128. Baltimore, MD, 2005.

[56] M. Szpyrka, B. Jasiul, K. Wrona, and F. Dziedzic. Telecommunications networks risk assessment with Bayesian networks. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pages 277–288. Springer, 2013.

[57] O. Stan, R. Bitton, M. Ezrets, M. Dadon, M. Inokuchi, O. Yoshinobu, Y. Tomohiko, Y. Elovici, and A. Shabtai. Extending attack graphs to represent cyber-attacks in communication protocols and modern IT networks. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[58] C.-W. Ten, C.-C. Liu, and M. Govindarasu. Vulnerability assessment of cybersecurity for scada systems using attack trees. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–8. IEEE, 2007.

[59] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of business process management*, volume 1. Springer, 2017.

[60] Avital Koren. Risk Management in ISO 9001. `https://isoupdate.com/resources/risk-management-in-iso-9001/`, 2019. Accessed: 15-09-2021.

[61] S. Adepu, N. K. Kandasamy, and A. Mathur. Epic: An electric power testbed for research and training in cyber physical systems security. In *Computer Security*, pages 37–52. Springer, 2018.

[62] A. Siddiqi, N. O. Tippenhauer, D. Mashima, and B. Chen. On practical threat scenario testing in an electric power ICS testbed. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, pages 15–21, 2018.

[63] ENISA. Communication network dependencies for ICS/SCADA Systems, 2016.