Using Bayesian Networks for a Cyberattacks Propagation Analysis in Systems-of-Systems

Jamal EL HACHEM*, Ali Sedaghatbaf†, Elena Lisova†, Aida Čaušević†
*Université Pau & Pays Adour, LIUPPA
{jamal.elhachem}@univ-pau.fr
†Mälardalen University
{ali.sedaghatbaf, elena.lisova, aida.causevic}@mdh.se

***Abstract*—System of Systems (SoS) represent a set of independent Constituent Systems (CS) that collaborate in order to provide functionalities that they are unable to achieve independently. We consider SoS as a set of connected services that needs to be adequately protected. The integration of these independent, evolutionary and distributed systems, intensifies SoS complexity and emphasizes the behavior uncertainty, which makes an SoS security analysis a critical challenge. One of the major priorities when designing SoS, is to analyze the unknown dependencies among CS services and vulnerabilities leading to potential cyberattacks. The aim of this work is to investigate how Software Engineering approaches could be leveraged to analyze the cyberattack propagation problem within an SoS. Such analysis is essential for an efficient SoS risk assessment performed early at the SoS design phase and required to protect the SoS from possibly high impact attacks affecting its safety and security. In order to achieve our objective, we present a model-driven analysis approach, based on Bayesian Networks, a sensitivity analysis and Common Vulnerability Scoring System (CVSS) with aim to discover potential cyberattacks propagation and estimate the probability of a security failure and its impact on SoS services. We illustrate this approach in an autonomous quarry example.**

***Index Terms*—Systems-of-Systems, Service Oriented Architectures, Bayesian Networks, Cyberattacks.**

I. Introduction

Software systems complexity has been systematically increasing to deliver services in diverse domains, such as autonomous vehicles, smart cities, defense, and E-health. These modern solutions are application domains of a specific type of systems called System-of-Systems (SoS). An SoS consist of a set of distributed systems called Constituent Systems (CSs) that interact to accomplish higher functionalities that none of the CSs is able to deliver in isolation [1]. In our work, we assume CSs to be services, as well as communication between CSs as a service per se. Since reliable and predictable communication enables correct functioning of an SoS, communication as a service is one of the main assets to consider [2].

These services are characterized by their managerial and operational independence, their geographic distribution, evolutionary development, and their emergent interactions leading to possibly (un)expected emergent behaviors [3]. These characteristics intensify SoS complexity and introduce significant challenges into SoS engineering, such as assuring the quality of system properties, in particular security engineering [4]. In this paper we focus on cyberattacks enabled by emergent behaviours among other factors.

The past few years, SoS solutions have suffered from severe security cyberattacks affecting SoS services, nation's economic plannings, and more importantly people's safety. Wannacry, Uber breach[1], moving jeep hack[2], Stuxnet[3], and Google building attack[4] are examples of recent high-impact cyberattacks targeting different SoS application domains. Many of these cyberattacks are results of exploiting a sequence of vulnerabilities triggered in different CSs and connected through service dependencies. Each of these vulnerabilities could be initially judged as a low impact for the CS it is identified at. However, the unknown sequences of triggered vulnerabilities connected at the SoS level through CS service dependencies intensifies the cyberattack and aggravate its effect.

The aim of this study is to address an SoS security analysis challenge while considering SoS specific characteristics. We focus on addressing *how to analyze SoS architectures to enable an appropriate analysis and quantitative evaluation of cyberattacks propagation at the SoS level*. This challenge should be addressed early at the design stage of the SoS development lifecycle, to reduce time and cost in case of late changes and protect SoS from cyberattacks targeting its services. Model Driven Engineering (MDE) has emerged in recent years as a key solution to address security-by-design challenges through the development of dedicated languages. Moreover, MDE offers a model transformation mechanism to automate the analysis of several SoS security architecture alternatives, to discover potential cyberattacks, analyze their propagation and take corrective actions until reaching a required security level.

Bayesian networks (BN) and a sensitivity analysis play a fundamental role in a quantitative analysis of SoS complexity and its behavioral aspects. BN are also effective in investigating CS dependencies and security emergent behaviors via cyberattacks probability, propagation and impact [5], [6].

Therefore, in this study we present a MDE approach based on a service decomposition, BN and a sensitivity analysis to address the SoS cyberattacks propagation analysis challenge.

[1]https://money.cnn.com/2017/12/18/technology/biggest-cyberattacks-of-the-year/index.html
[2]https://www.youtube.com/watch?v=ysAam9Zmdv0
[3]http://large.stanford.edu/courses/2015/ph241/holloway1/
[4]https://www.wired.com/2013/05/googles-control-system-hacked/

We use the SoSSec approach [7] to model an SoS architecture, and propose a mapping of such a model to BNs. Using BNs allows us to get a quantitative assessment for which initial probability estimations are required. The ways of getting such estimations are discussed in this work as well. Moreover, a sensitivity analysis of the results from a BN model analysis, helps to interpret the actual meaning of derived probabilities of the services being attacked in an SoS context. We illustrate the proposed approach on an autonomous quarry example, modeled as a complex SoS.

The rest of the paper is organized as following. Section II presents the background. Section III introduces an autonomous quarry as our running example. The details of our approach are presented in Section IV, whereas analysis results are demonstrated in Section V. Section VI concludes the paper.

II. Background

A. Security terminology

Security can be defined as a system property allowing a system "to perform its mission or critical functions despite risks posed by threats" [8], where a threat can be defined as "the potential source of an adverse event" [8]. A vulnerability is a flow in the system that can be exploit by an adversary that targets one of the system assets, i.e., things needs to be protected. A concrete threat realization is an attack, where a cyberattack is an attack performed via a cyberspace.

B. Investigating Existing Approaches for SoS Security Architecture Analysis

Several works have been proposed to address an SoS analysis, but only few consider SoS security. These can be divided in the following categories.

Goal-based approaches: In [9] the authors propose dynamic Agent Based Game Theoretic simulations to analyze smart grids information security. Although the proposed approach seems promising to calculate the probability of successful attacks at a specific time of a scenario simulation, the attacker/defender model, in its current form, is too simple to represent an SoS with its CSs and their interactions. Moreover, the analysis does not reveal specific information on the vulnerabilities and how they are triggered, leading to the successful elementary attacks. [10] present an agent-based modeling language to analyze threats at the requirement level of the SoS development. The analysis is based on a set of identified threatening event and assumptions rules over goal decomposition trees which make it limited to the specified rules. Moreover, the paper does not show details on how the rules are selected and defined. In [11] authors propose a simulation approach based on multi-agent systems to simulate and analyze cascading attacks. The presented method seems promising, however the analysis is qualitative which is not helpful to guide the architect decisions.

Threat-based approaches: In [12] authors discuss the challenges towards safe and secure SoS. They propose to use the system theoretic safety analysis method to systematically uncover possible undesired losses in SoS. This work is at its early stages and authors note that their method needs further development to be applicable to SoS security analysis. Authors of [13] present a framework for formal SoS threat analysis. They study the impact of threats that can arise from adding or removing assets in an SoS scenario. The proposed approach for run-time analysis mainly considers the evolutionary characteristic of an SoS. However, the approach analyzes the distribution of known threats/vulnerabilities over the CSs at run-time, not those triggered and linked due CSs interactions and dependencies.

Graph-based approaches: [14] presents a use-case scenarios based approach, for safety and security consideration in smart homes. The approach uses attack trees and safety use cases to harmonize the security structure to the safety requirements. This approach allows risk assessment related to a specific SoS application domain and based on defined attack scenarios. However, the idea lacks deeper investigation and corresponding tools. In [15], authors extend the Functional Dependency Network Analysis (FDNA) to make it applicable for an SoS to analyze the impact of cyberattacks on the SoS interdependencies. In this study, security is barely considered by adding a weight indicating the availability of data to model the effect of an attack. There are no security concepts describing the vulnerabilities or the attack, neither the SoS emergent behavior. A similar study [16] introduces a general framework to assess the security risk of a single security incident on multiple CSs. FDNA is used to measure the SoS ability to operate effectively if one or more CS fail.

The presented studies touch on security only in a broad way. Most of them do not explicitly or clearly consider the SoS specific characteristics in the analysis. Moreover, they lack pertinent details to analyze SoS interactions, dependencies and emergent behavior, as well as to support the early discovery and anticipation of unforeseen security problems.

C. Bayesian Networks for a Cyberattacks Analysis

Among the probabilistic graph models approaches, Bayesian Betworks (BN) gained a noticeable interest as a suitable approach to express the conditional dependencies between random variables, e.g. CS vulnerabilities and dependencies, therefore BN are applicable to address the cyberattacks propagation analysis challenge.

BN are probabilistic directed acyclic graphs that have been effectively used in various areas, e.g., medical diagnosis [17], reliability engineering [18] and system troubleshooting [19]. However, the area of SoS security analysis has not gained much attention. BNs are a powerful tool for representing causal dependencies and reasoning under uncertainties. This makes them an appropriate means to analyze cyberattacks propagation.

In a BN, each node represents a random variable and each arc indicates a probabilistic dependency between two variables. If there is an arc from node X to node Y , X is called the parent of Y, and any change in X may affect the value of *Y*. Nodes without parents are called root nodes, and nodes without children are called leaf nodes. To each root node a prior probability table (PPT) is associated, which assigns a probability to each value that the random variable may take. To propagate changes from the root nodes to other nodes in the graph, a probability function is associated to each non-root node. This function takes as input the set of possible values for parent nodes and gives as output the probabilities of possible values for the variable represented by the node itself. The relation between input and output values is represented by a conditional probability table (CPT).

For any subset of BN nodes, we can define a joint probability distribution, which can be computed using the following formula (1). We can estimate the probability distribution of any none-root node by marginalizing the joint distribution with respect to the variable represented by that node [20].

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|parents(X_i)) \qquad (1)$$

Compared to traditional attack modeling methods e.g., attack trees (AT), BNs have attractive features for representing complex dependencies, such as:

1) Logic gates in ATs (e.g., AND gates and OR gates) can only represent simple and deterministic dependencies among nodes, whereas CPTs can represent complex and probabilistic dependencies with deterministic dependency being just a special case (i.e., probability 1 or 0);
2) In an AT, each node has a binary state, while BN allows us to consider multiple security states for each node;
3) ATs cannot model dependencies among nodes at the same level, whereas BNs impose no limitations on dependency ends.

The last feature is of special importance for analyzing the propagation of cyberattacks.

III. An autonomous quarry SoS

In this paper we use an example of an autonomous quarry consisting of the following: *(1)* a remote control room with a human operator overviewing all processes and having a possibility to take over control if needed; *(2)* a fleet of fully autonomous carriers, that move the load, e.g., stones of different granularity; *(3)* a carrier charging station; *(4)* a stone extraction point at which a wheel loader loads the carrier with stones; *(5)* a point where larger stones are crashed into smaller ones; *(6)* a storage facility where carriers deliver lower granularity stones. The decision making part is placed in both the control room and locally at carriers. In a normal operation mode the carrier just follows the receiving commands, as the control room keeps the communication with carriers, updating them on changes in the quarry (i.e., routes, other carriers locations and statuses), current tasks to be done, available maneuver space, and the allowed speed.

As a quarry is a harsh environment, quality of communication with the control can degrade. Therefore, a carrier is equipped with sensors to enable localization of objects in a close proximity, detection of their speed and making its own decision regarding required actions even if it overrules the command from the control room, e.g., an emergency stop when an object has been detected dangerously near by. To overrule the prescribed action and switch to a self-control mode, a carrier has to make a decision about a communication channel with the control room being unreliable enough and detect a condition requiring an emergent response. After this the carrier executes one of the failing safe scenarios. The remote control room might as well broadcast an emergency stop message as a safety measure to react upon a detected hazardous situation. Due to the harsh environment in the quarry, additional techniques like relaying can be used to increase the reliability of a broadcast communication to an acceptable level [21].

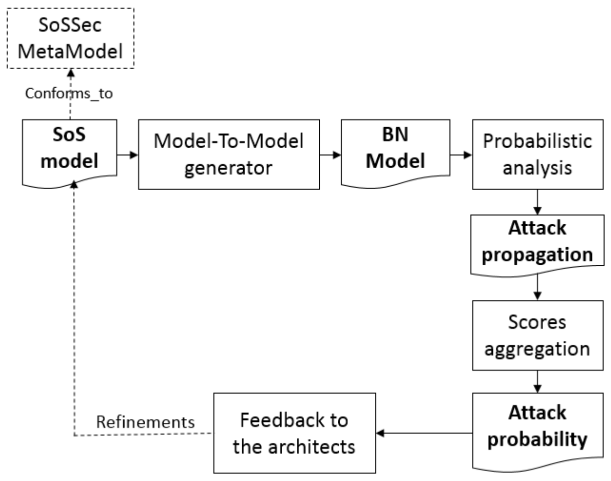

Fig. 1: The proposed approach

We consider the following types of communication to exist in the quarry [2]: *(1)* point-to-point (p2p) communication between the control room and every vehicle in the quarry, to transmit the control information about vehicle's actions and relevant information about current status of other vehicles and quarry in general; *(2)* broadcast communication from the control room that propagates the emergency stop message to all vehicles whenever needed; *(3)* broadcast communication from each vehicle with a smaller range than type *(2)* communication, as it's main function is to broadcast a status information regarding vehicles in the quarry to increase awareness of vehicles in the close proximity. The functionality is redundant in the normal operational mode.

IV. Bayesian networks for an attack propagation analysis in SoS

Fig. 1 depicts the methodology proposed in this paper used for analyzing SoS and can be synthesized as follows: *(1)* the SoS to be analyzed is modelled using the SoSSec modeling language and graphical editor; *(2)* the resulting SoS security architecture is transformed into a BN model; *(3)* the obtained BN model is analyzed to discover the potential cyberattacks and estimate the probability of a security breach and quantitatively assess its impact on SoS services. In the following sub-sections we detail each step.

A. *SoS architectures modeled using the SoSSec Method*

To properly analyze an SoS including security, we need well defined models representing the SoS security architectures [22]. To achieve this we choose to use the SoSSec approach [7]. SoSSec extends the System Modeling Language (SysML) to model the structure of an SoS considering its characteristics (i.e., CSs, interfaces, global and individual goals) and its behavior (i.e., interactions, operations, exchange of data). SoSSec allows description of *known vulnerabilities* for each CS, including the conditions that trigger these vulnerabilities as well as those resulting from a triggered vulnerability, respectively denoted by pre-, and post-condition(s).

Once the SoS security architecture is modeled, the next step is to perform its analysis to discover potential sequences of triggered vulnerabilities possibly exploited by cyberattacks, and formally evaluate their probability and impact. This analysis results provide an useful feedback and recommendations that could be sent to the security architects, helping them improve the SoS security architecture and avoid discovered cyberattacks. One of the main advantages of choosing the SoSSec method as the basis for our cyberattacks analysis is the fact that SoSSec is built following the MDE principles and guarantee an automated mapping between the modeling and analysis phases, thereby allowing an iterative analysis of several SoS security architectures until reaching an acceptable security level.

Fig. 2 and Fig. 3 depict an autonomous quarry SoS architecture. Based on the scenario introduced in Section III, Fig. 2 shows the SoSSec structural Block Diagram (BD) illustrating two CSs: the *Remote_Control_Room_CS* and the *Vehicle_Control_Room_CS*. Each of these CS are managed by an independent organisation (SoS managerial independence characteristic) and have its own operations (SoS operational independence characteristic). Fig. 3, on the other hand, shows the SoSSec behavioral Activity Diagram (AD) illustrating an autonomous quarry scenario, in particular the CS interactions. For each CS, a list of potential vulnerabilities is defined by the security architects and analysts. For our autonomous quarry scenario we studied existing reviews, conference papers and and reports/press news such as [23]–[26] to assign the list of vulnerabilities shown in red.

Each vulnerability is enriched by a list of pre-, and post-conditions representing constraints that trigger a vulnerability and the results of a triggered vulnerability. The vulnerability details are extracted from the Common Vulnerabilities and Exposures (CVE) catalog. CVE is a list of common identifiers for known security vulnerabilities, result of the international cyber security community effort, and is considered as the industry standard for vulnerability and exposure names. For each vulnerability and exposure, CVE offers a standardized description allowing the extraction of such information as a vulnerability type, its pre-, and post-conditions.

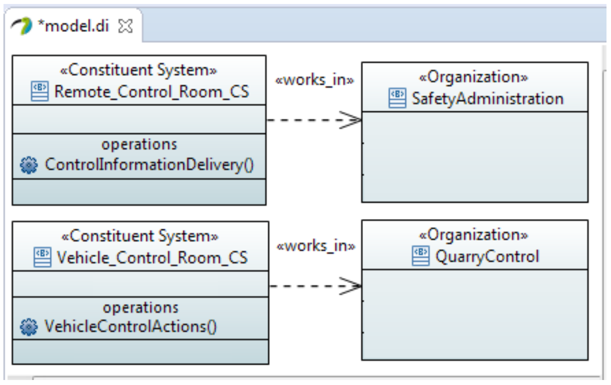

Fig. 2: An autonomous quarry block diagram modeled using the SoSSec graphical editor

B. *Mappings from SoSSec to Bayesian Networks*

To perform an analysis of the SoS emergent cyberattacks exploiting service interactions and communication, the approach should include the properties described in models [27], [28]. SoSSec supports specification of known vulnerabilities and their pre-, and post-conditions. However, the sequences of vulnerabilities exploited through the service interactions and dependencies, their success probability and their impact on SoS security remain unknown. The complexity and heterogeneity inherited in an SoS makes this task even more challenging. In order to reason about a security level of an SoS architecture and decide about appropriate attack countermeasures, there is a need for a formalism that can capture this uncertainty. BNs can graphically model the dependencies among vulnerabilities and services, and provide a probabilistic results to reason about the impact of different attack scenarios to SoS security.

Therefore, in the second step of our approach, the MDE model-to-model technique is used to semi-automatically generate a BN corresponding to a SoS architecture modeled using SoSSec. Algorithm 1 describes the BN models generation procedure. The generated BN has four conceptual levels (see Fig. 4). At the highest level, we have a single leaf node which value indicates whether SoS security is compromised or not. Each node at the second level corresponds to a CS in the SoSSec model. There is an arc from each node at this level towards the leaf node, since security of each CS directly affects SoS security. The third level is dedicated to the services provided by each CS. Each service has one outgoing arc to its encompassing CS. Also, there is an arc from a service node A to a service node B if at least one of the post-conditions of the node A is the same as one of the pre-conditions of the node B in one of the Activity Diagrams (ADs). At least one vulnerability is associated to each of the services at the third level in the SoSSec AD. These vulnerabilities are presented in the fourth level of the generated BN. Similar to service-level connections, vulnerability nodes may be connected to each other or to services if they have common pre-, or post-conditions.

The BN derived from the SoSSec model of an autonomous quarry is depicted in Fig. 5. The BN has two nodes at the

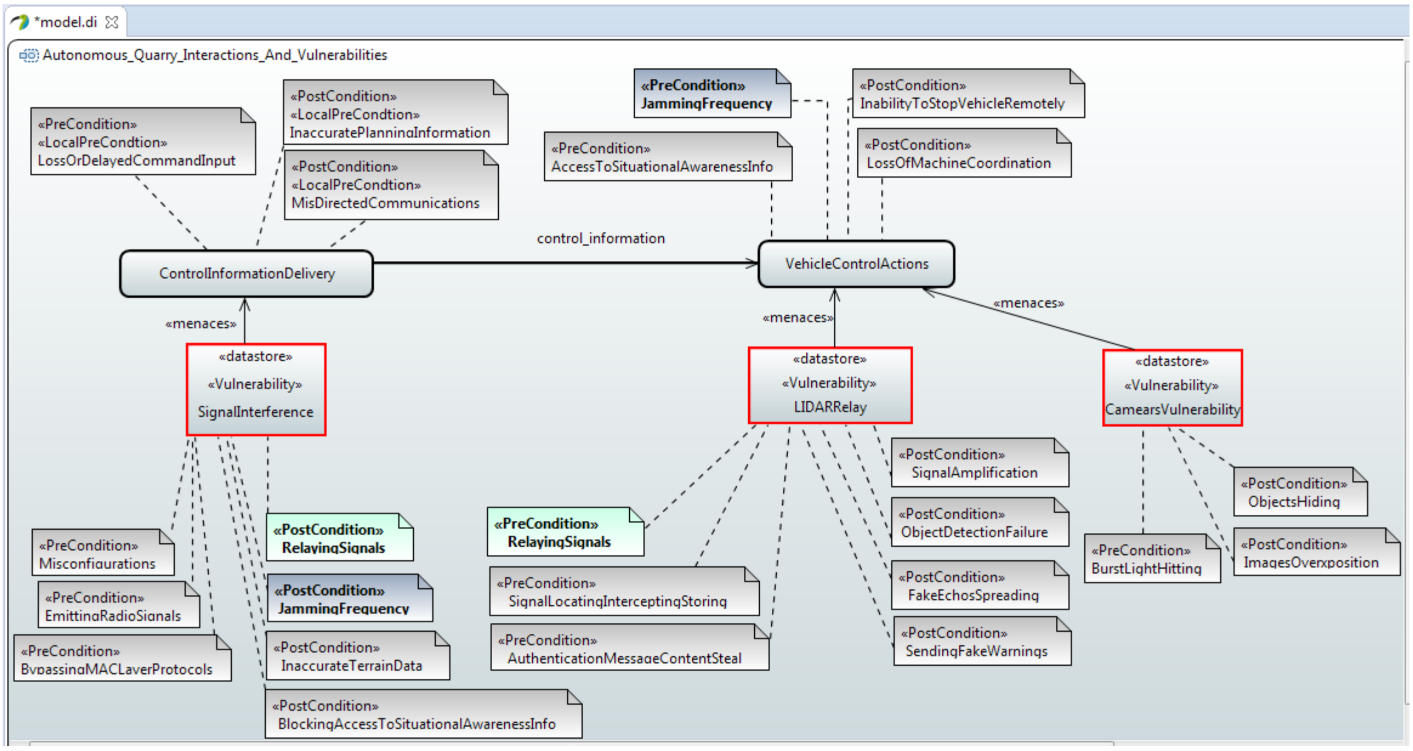
*model.di
Autonomous_Quarry_Interactions_And_Vulnerabilities
«PreCondition» «LocalPreCondtion» LossOrDelayedCommandInput
«PostCondition» «LocalPreCondtion» InaccuratePlanningInformation
«PostCondition» «LocalPreCondtion» MisDirectedCommunications
«PreCondition» JammingFrequency
«PreCondition» AccessToSituationalAwarenessInfo
«PostCondition» InabilityToStopVehicleRemotely
«PostCondition» LossOfMachineCoordination
control_information
ControlInformationDelivery
VehicleControlActions
«menaces»
«menaces»
«menaces»
«datastore» «Vulnerability» SignalInterference
«datastore» «Vulnerability» LIDARRelay
«datastore» «Vulnerability» CamearsVulnerability
«PreCondition» Misconfiqurations
«PreCondition» EmittingRadioSignals
«PreCondition» BypassingMACLayerProtocols
«PostCondition» RelayingSignals
«PostCondition» JammingFrequency
«PostCondition» InaccurateTerrainData
«PostCondition» BlockingAccessToSituationalAwarenessInfo
«PreCondition» RelayingSignals
«PreCondition» SignalLocatingInterceptingStoring
«PreCondition» AuthenticationMessageContentSteal
«PostCondition» SignalAmplification
«PostCondition» ObjectDetectionFailure
«PostCondition» FakeEchosSpreading
«PostCondition» SendingFakeWarnings
«PostCondition» ObjectsHiding
«PreCondition» BurstLightHitting
«PostCondition» ImagesOverposition

Fig. 3: An autonomous quarry activity diagram modeled using the SoSSec graphical editor

second level corresponding to the two CSs in our SoSSec model (i.e., *VehicleControlRoom* and *RemoteControlRoom*). According to the AD in Fig. 3, each of these CSs provides only one service. Therefore, the third level of the BN model includes two nodes. The arc between these nodes indicates the possibility of an attack propagation from *CtrlInfoDelivery* to *CtrlAction*. The fourth level includes three vulnerability nodes derived from the AD. Since *RelayingSignals* is both a pre-condition of *LIDARRelay* and a post-condition of *SignalInt.*, there is an arc from *SignalInt.* to *LIDARRelay* at the fourth level.

C. Security Evaluation

In order to evaluate SoS security, each node of the BN model generated in step two, needs to be enriched with appropriate estimations for the PPT/CPT values. In the following sub-sections, we elaborate how to provide these estimations and incorporate them in the BN analysis.

1) PPT Estimation: Each root node in the BN generated by Algorithm 1, has assigned PPT that includes a single entry denoting an exploitation probability of the vulnerability modeled for that node. For scoring vulnerabilities, there is a considerable research done, e.g., Common Vulnerability Scoring System (CVSS)[5] is a well-known and standard scoring system for estimating the severity of network vulnerabilities. There are also other scoring systems, e.g., DREAD [29] and OWASP [30].

[5]https://www.first.org/cvss/v3.0/specification-document

In this paper, we propose a group of basic metrics inspired by the CVSS system to score vulnerabilities. We select the following metrics, as described in Table I, suitable to help security analysts to produce scores that are consistent and arguable across various situations in the context of our work.

CVSS is composed of three metric groups, Base, Temporal, and Environmental, each consisting of a set of metrics. In this work, we focused on the Base metric group representing the intrinsic characteristics of a vulnerability that are constant over time and across user environments. It is composed of two sets of metrics: 1) The Exploitability metrics (*Attack Vector, Privileges Required, User Interaction* and *Scoop* in table I) reflect the ease and technical means by which the vulnerability can be exploited; 2) The Impact metrics (*Confidentiality Impact, Integrity Impact* and *Availability Impact* in table I) reflect the direct consequence of a successful exploit, and represent the consequence to the thing that suffers the impact, which we refer to formally as the impacted component. When the Base metrics are assigned values by an analyst, the Base equation computes a CVSS score ranging from 0.0 to 10.0. In order to convert this score to a PPT entry, we simply divide it by 10 since a PPT entry is a probabilistic value between zero and one. For example, 10, 8.2 and 7.5 are the CVSS scores calculated for Signal Interference, LIDDAR Relay and Camera Vulnerability in the Autonomous Quarry example. Their corresponding PPT value would be 1, 0.82 and 0.75.

2) CPT Estimation: Providing estimations for CPT values is more difficult than for PPTs, since there is no well-known metric for quantifying the dependencies among vulnerabilities,

```
$SysNode \leftarrow new\ BNNode();$
for each CS block cs in SoSSec BD do
    $CSNnode \leftarrow new\ BNNode();$
    $SysNode.addParent(CSNode);$
    for each service block sv in SoSSec BD do
        $SVNode \leftarrow new\ BNNode();$
        $CSNode.addParent(SVNode);$
    end
end
for each AD ad in SoSSec model do
    for each service sv in ad do
        for each vulnerability vl associated to sv do
            $VLNode \leftarrow new\ BNNode();$
            $SVNode.addParent(VLNode);$
        end
    end
    for each pair (sv, vl) of services and vulnerabilities
     in ad do
        let $SPre$ be the set of pre-conditions of $sv$;
        let $SPost$ be the set of post-conditions of $vl$;
        if $SPre \cap SPost \neq \emptyset$ then
            $SVNode.addParent(VLNode);$
        end
    end
    for each pair (vl1, vl2) of vulnerabilities in ad do
        let $VPre$ be the set of pre-conditions of $vl1$;
        let $VPost$ be the set of post-conditions of $vl2$;
        if $VPre \cap VPost \neq \emptyset$ then
            $VL1Node.addParent(VL2Node);$
        end
    end
end
```

Algorithm 1: A BN generation algorithm

services and corresponding CSs. In this case, it is possible to rely on security experts' knowledge, who may suggest proper values based on their experience from past incidence reports and their knowledge about each service/CS functionality.

3) BN Analysis: Based on the input PPT/CPT values, a security failure probability for each service/CS and the whole SoS can be estimated. In the BN model generated in step two, let us assume that $X_{(l,n)}$ denotes the nth node at the lth level, and $PAR(X_{(l,n)})$ denotes the set of parent nodes of $X_{(l,n)}$. For example, for node $X_{(2,2)}$ in Fig. 4, we have $PAR(X_{(3,1)}) = \{X_{(3,2)}, X_{(4,1)}, X_{(4,2)}\}$. Using Eq. 1, we can calculate the joint probability of each node and its parent nodes. For example, the joint probability of $X_{(3,1)}$ and its parents can be derived as:

$$P(X_{(3,1)}, X_{(4,1)}, X_{(4,2)}) = P(X_{(3,1)}|X_{(4,1)}, X_{(4,2)}). P(X_{(4,1)}, X_{(4,2)}) \quad (2)$$

where $P(X_{(4,1)}, X_{(4,2)})$ can be decomposed recursively. Now, based on the joint probability, the marginal probability for each node can be obtained using the following:

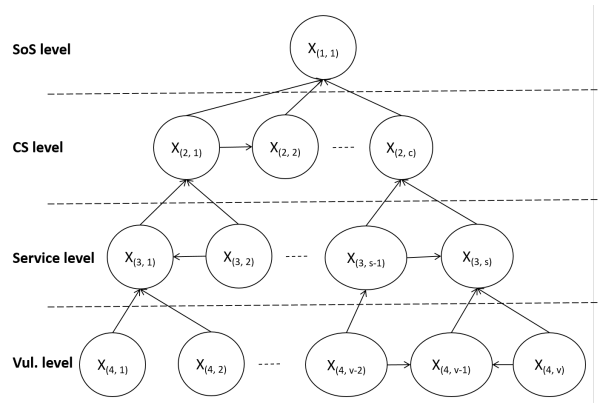

Fig. 4: A multi-level BN

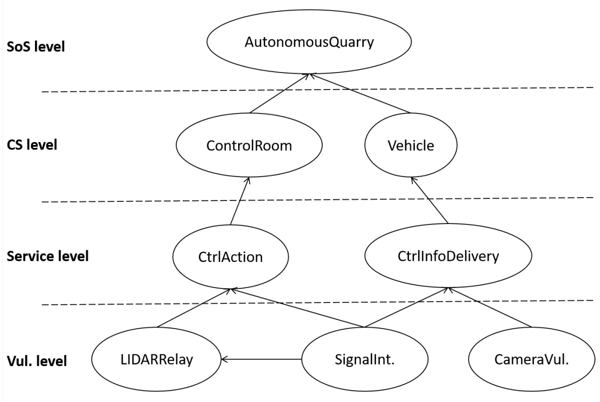

Fig. 5: A BN model of autonomous quarry

$$P(X_{(l,n)}) = \sum_{PAR(X_{(l,n)})} P(X_{(l,n)}, PAR(X_{(l,n)})) \quad (3)$$

$P(X_{(l,n)})$ can be estimated for each non-root node using Eq. 3. The estimated value expresses the probability of a security failure for the corresponding service/CS in the SoS (or the whole SoS if it is related to the leaf node).

V. Analyzing an Attack Propagation in an Autonomous Quarry SoS

A. Initial Evaluations

To assess applicability of our methodology, we have analyzed the BN model of an autonomous quarry presented in Fig. 5. The PPT values evaluated using CVSS scores and the CPT values suggested by security experts, are presented in Fig. 6. Using these values and Eq. 3 we have estimated the probability of a security failure for service, CSs and the whole SoS. The estimations in Tab. II indicate that *CtrlAction* is more vulnerable than *CtrlInfoDel.* and *CtrlRoom* is more affected by the vulnerabilities and their propagation than *Vehicle*.

Using this approach, we are able to estimate and compare security levels of different CSs and services. Also we can analyze and compare an impact of removing vulnerabilities to security of the SoS, its CSs and their services. For example,

TABLE I:
SIGNAL INTERFERENCE VULNERABILITY SCORE CALCULATION USING CVSS BASE SCORE CALCULATOR

Metric	**Value**	**Comments**
Attack Vector	Network	Signal Interference vulnerability is remotely exploitable by transmitting a high-range signal for example
Attack Complexity	Low	The attacker doesn't need a specialized access conditions and can expect repeatable attack success against the physical transmission of signals during a communication process e.g. through situations such as noise, interference and collision
Privileges Required	None	The attacker is unauthorized prior to attack, consequently he does not require any access to settings or files to achieve his attack
User Interaction	None	The physical support for signal transmission can be exploited without interaction from any user
Scope	Changed	The Signal Interference exploited vulnerability can affect the situational awareness info as well as the signals transmitted to another system component
Confidentiality Impact	None	There is no confidentiality loss within the impacted component
Integrity Impact	High	Exploiting this vulnerability results in a loss of integity and/or protection of the data transmitted on the wireless medium e.g. through noise, interference, collision
Availability Impact	High	After exploiting this vulnerability the attacker is able to deny access to resources in the impacted component such as blocking access to situational awareness information

current evaluations indicate that the probability of a security breach is 0.7659 for an autonomous quarry. Assuming that the camera has no vulnerabilities, this probability would reduce to 0.4801, whereas removing the SignalInt. vulnerability would lead to a higher decrease (i.e., approximately 0.4411).

TABLE II:
BN ANALYSIS RESULTS

Element	Security Failure Probability
CtrlAction	0.9011
CtrlInfoDel.	0.8375
CtrlRoom	0.816
Vehicle	0.7619
AutonomousQuarry	0.7835

VI. Conclusion and Future Work

A. Sensitivity Analysis

We have demonstrated how the BN analysis might help in quantifying the effects of cyberattacks propagation to SoS security. However, reliability of the analysis results depends on the accuracy of the input parameters, i.e., PPTs and CPTs. Due to data scarcity, it is rather difficult to assure the accuracy of PPT/CPT entries. To address this issue, a sensitivity analysis is an adequate solution not only to detect the input parameters that have the highest effect on the accuracy of the analysis results, but also, to determine the robustness of the analysis results against small changes in PPT/CPT values.

For the autonomous quarry example, we varied the value of each PPT/CPT entry by 0.1 (while keeping others unchanged) and repeated the BN analysis step. For each PPT we recorded the amount of change in the SoS security failure probability.

	True	False
SignalInt.	1	0
CameraVul.	0.75	0.25

(a) SignlaInt. & CamVul. PPTs

SignalInt.	True	False
True	0.82	0.18
False	0.12	0.88

(b) CPT of LIDARRelay

CtrlAction	True	False
True	0.9	0.1
False	0.05	0.95

(c) CPT of CtrlRoom

CtrlInfoDel.	True	False
True	0.9	0.1
False	0.05	0.95

(d) CPT of Vehicle

SignalInt.	LIDARRel.	True	False
False	False	0	1
False	True	0.3	0.7
True	False	0.4	0.6
True	True	0.99	0.01

(e) CPT of CtrlAction

SignalInt.	CameraVul.	True	False
False	False	0	1
False	True	0.5	0.5
True	False	0.5	0.5
True	True	0.95	0.05

(f) CPT of CtrlInfoDelivery

CtrlRoom	Vehicle	True	False
False	False	0	1
False	True	0.6	0.4
True	False	0.4	0.6
True	True	1	0

(g) CPT of AQ

Fig. 6: Input estimations for autonomous quarry

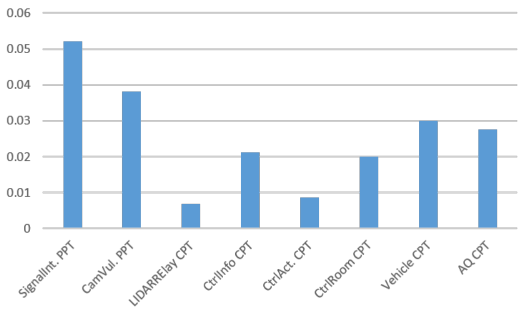

Fig. 7: Sensitivity analysis results

However, for CPTs the recorded value was the average of the changes made by all entries. Fig. 7 shows the recorded changes. Accordingly, sensitivity of the output accuracy to the probability of the SignalInt. vulnerability is more than the vulnerability in cameras. The analyst should be more careful about the CPTs of an autonomous quarry and Vehicle, since they have a higher impact on the output accuracy.

Given the complexity of SoS and the growth of their application, there is a need for a suitable security analysis that can capture dependencies between CSs. In this paper we present a semi-automatic approach for modeling and analysis of a cyberattack propagation in SoS. The foundation of our approach is MDE where we start from already existing SoSSec and compliment it with a semi-automatic transformation to BNs for analysis purposes. Given this we are able to estimate a probability of a security failure and compare security levels of different CSs in an SoS as well as to estimate how mitigation of a particular vulnerability affects the security level. For example, for the considered autonomous quarry mitigating a vulnerability related to signal interference is more effective from a security perspective compared to other analyzed vulnerabilities. As a future work, we aim to provide a complete tool support for the presented approach.

Acknowledgement

This work is supported by the Serendipity project funded by SSF, the SAFSEC-CPS project funded by KKS.

References

[1] M. Maier, "Architecting principles for SoS," *Systems Engineering*, vol. 1, pp. 267 – 284, 1998.
[2] E. Lisova, J. El Hachem, and A. Causevic, "Investigating attack propagation in a sos via a service decomposition," in *IEEE SERVICES Workshop on Cyber Security and Resilience in the Internet of Things*, July 2019.
[3] H. Kopetz, O. Höftberger, B. Frömel, F. Brancati, and A. Bondavalli, "Towards an understanding of emergence in SoS," in *System of Systems Engineering Conference, Texas, USA*, 2015, pp. 214–219.
[4] J. Dahmann, "System of systems pain points," *INCOSE International Symposium*, vol. 24, no. 1, pp. 108–121, 2014.
[5] S. Chockalingam, W. Pieters, A. Teixeira, and P. van Gelder, "Bayesian network models in cyber security: A systematic review," in *Secure IT Systems*. Springer International Publishing, 2017, pp. 105–122.
[6] Y. Zhou, C. Zhu, L. Tang, W. Zhang, and P. Wang, "Cyber security inference based on a two-level bayesian network framework," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 3932–3937.
[7] J. El Hachem, T. A. Khalil, V. Chiprianov, A. Babar, and P. Aniorte, "A model driven method to design and analyze secure architectures of systems-of-systems," in *22nd International Conference on Engineering of Complex Computer Systems, Fukuoka, Japan*, 2017, pp. 166–169.
[8] R. Kissel, *Glossary of key information security terms*. U.S. Dept. of Commerce, National Institute of Standards and Technology, 2006.
[9] R. Abercrombie and F. Sheldon, "Security analysis of smart grid cyber physical infrastructures using game theoretic simulation," in *IEEE Symposium Series on Computational Intelligence*, 2015, pp. 455–462.
[10] P. Meland, E. Paja, E. Gjære, S. Paul, F. Dalpiaz, and P. Giorgini, "Threat analysis in goal-oriented security requirements modelling," *International Journal on Secure Software Engineering*, vol. 5, no. 2, pp. 1–19, 2014.
[11] J. El Hachem, V. Chiprianov, V. V. Graciano Neto, and P. Aniorte, "Extending a multi-agent systems simulation architecture for systems-of-systems security analysis," in *2018 13th Annual Conference on System of Systems Engineering*, 2018, pp. 276–283.
[12] A. Kobetski and J. Axelsson, "Towards safe and secure systems of systems: Challenges and opportunities," in *Proceedings of the Symposium on Applied Computing*, ser. SAC, 2017, pp. 1803–1806.
[13] M. Mori, A. Ceccarelli, T. Zoppi, and A. Bondavalli, "On the impact of emergent properties on SoS security," in *11th System of Systems Engineering Conference, Kongsberg, Norway*, 2016, pp. 1–6.
[14] J. Nicklas, M. Mamrot, P. Winzer, D. Lichte, S. Marchlewitz, and K. Wolf, "Use case based approach for an integrated consideration of safety and security aspects for smart home applications," in *11th System of Systems Engineering Conference, Kongsberg, Norway*, 2016, pp. 1–6.
[15] C. Guariniello and D. DeLaurentis, "Communications, information, and cyber security in systems-of-systems: Assessing the impact of attacks through interdependency analysis," *Procedia Computer Science*, vol. 28, pp. 720 – 727, 2014.
[16] J. Dahmann, G. Rebovich, M. McEvilley, and G. Turner, "Security engineering in a system of systems environment," in *IEEE Systems Conference*, 2013.
[17] Y. S. Chang, C. T. Fan, W. T. Lo, W. C. Hung, and S. M. Yuan, "Mobile cloud-based depression diagnosis using an ontology and a bayesian network," *Future Generation Computer Systems*, vol. 43, pp. 87–98, 2015.
[18] J. Luque and D. Straub, "Reliability analysis and updating of deteriorating systems with dynamic bayesian networks," *Structural Safety*, vol. 62, pp. 34–46, 2016.
[19] D. Lang, P. Wunderlich, M. Heinz, L. Wisniewski, J. Jasperneite, O. Niggemann, and C. Röcker, "Assistance system to support troubleshooting of complex industrial systems," in *14th IEEE International Workshop on Factory Communication Systems*.
[20] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.
[21] S. Girs, E. Uhlemann, and M. Björkman, "Increased reliability or reduced delay in wireless industrial networks using relaying and luby codes," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, Sep. 2013, pp. 1–9.
[22] J. El Hachem, Z. Pang, V. Chiprianov, A. Babar, and P. Aniorte, "Model Driven Software Security Architecture of Systems-of-Systems," in *APSEC*, Hamilton, New Zealand, 2016, pp. 89–96.
[23] V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," *Computer Communications*, vol. 132, pp. 17 – 41, 2018.
[24] D. Elliott, W. Keen, and L. Miao, "Recent advances in connected and automated vehicles," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 2, pp. 109 – 131, 2019.
[25] N. Sousa, A. Almeida, J. Coutinho-Rodrigues, and E. Natividade-Jesus, "Dawn of autonomous vehicles: review and challenges ahead," *Proceedings of the Institution of Civil Engineers - Municipal Engineer*, vol. 171, no. 1, pp. 3–14, 2018.
[26] "The white-hat hacking machine: Meet mayhem, winner of the darpa contest to find and repair software vulnerabilities," *IEEE Spectrum*, vol. 56, no. 2, pp. 30–35, Feb 2019.
[27] *INCOSE Systems Engineering Body of Knowledge, version 1.6*. INCOSE UMS, March 2016.
[28] N. Medvidovic and R. Taylor, "A classification and comparison framework for software architecture description languages," *IEEE Transactions on Software Engineering*, vol. 26, no. 1, pp. 70–93, 2000.
[29] J. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, and A. Murukan, *Improving web application security: threats and countermeasures*. Microsoft Corporation Washington, DC, 2003.
[30] OWASP, "The open web application security project," https://www.owasp.org/, 2019.